

(19) 日本国特許庁 (JP)

## (12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平 10-21134

(43) 公開日 平成10年(1998)1月23日

(51) Int. Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F	12/00	5 4 5	G 0 6 F	12/00 5 4 5 A
		5 1 4		5 1 4 K
	3/06	3 0 1		3/06 3 0 1 M
	13/00	3 5 7		13/00 3 5 7 Z
審査請求 未請求 請求項の数 1 2 O L			(全 4 1 頁)	

(21) 出願番号 特願平8-174561

(22) 出願日 平成8年(1996)7月4日

(71) 出願人 000005049

シャープ株式会社

大阪府大阪市阿倍野区長池町22番22号

(72) 発明者 土居 克良

大阪府大阪市阿倍野区長池町22番22号 シ

ャープ株式会社内

(72) 発明者 竹澤 創

大阪府大阪市阿倍野区長池町22番22号 シ

ャープ株式会社内

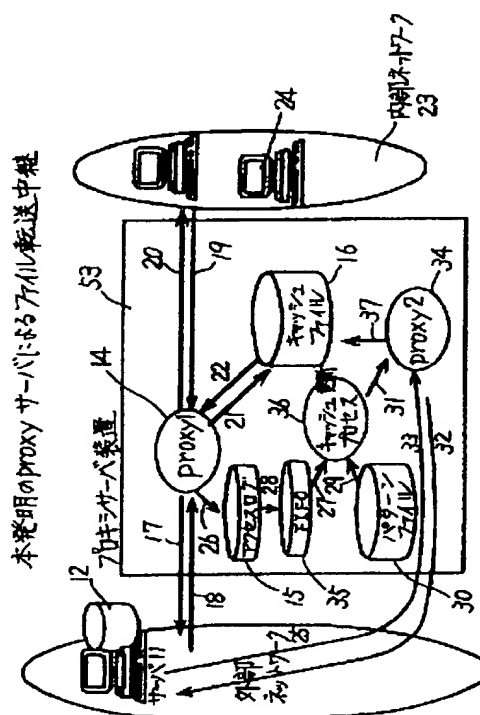
(74) 代理人 弁理士 深見 久郎

(54) 【発明の名称】 ゲートウェイ装置、クライアント計算機およびそれらを接続した分散ファイルシステム

## (57) 【要約】

【課題】 キャッシュファイルを新鮮に保ちながら、キャッシュヒット率の向上したゲートウェイ装置を提供すること。

【解決手段】 取得したファイルオブジェクトをキャッシュファイル16に格納し、アクセスログ15を格納する第1のproxyプロセス14に加えて、アクセスログ15から所定の規則に従ってアクセスログを抽出するキャッシュ更新プロセス36と、この抽出されたアクセスログのファイルオブジェクトをサーバ計算機11から取得してキャッシュファイル16に格納する第2のproxyプロセス34を含む。キャッシュ更新プロセス36は、第1のproxyプロセス14がキャッシュにヒットした場合にそのキャッシュにヒットしたファイルオブジェクトのアクセスログに基づいて第2のproxyプロセス34がキャッシュ内容を更新するとさらに効果的である。



## 【特許請求の範囲】

【請求項1】 クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するように設けられるゲートウェイ装置であって、

前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に応答して、当該ファイルオブジェクトにより指定される前記第2のネットワークのサーバ計算機に対して、当該ファイルオブジェクトに対するアクセス要求を行なうネットワーク

ファイル中継手段を含み、

前記ネットワークファイル中継手段は、取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、

過去の一定期間内に行なわれたファイルオブジェクトの転送記録を蓄積するファイル転送記録手段と、

ファイルオブジェクトに対するアクセス要求に答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含み、

前記ゲートウェイ装置はさらに、前記ファイル転送記録手段から所定の規則に従って転送記録を抽出し、当該転送記録のファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納するためのキャッシュ更新手段を含むゲートウェイ装置。

【請求項2】 前記キャッシュ更新手段はさらに、特定の文字列パターンを予め記憶するための文字列パターン記憶手段を含み、

前記キャッシュ更新手段は、前記特定の文字列パターンと一致するファイルオブジェクト名称を含む転送記録を前記ファイル転送記録手段から抽出し、当該ファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納する、請求項1記載のゲートウェイ装置。

【請求項3】 前記ネットワークファイル中継手段は、各々が第1のネットワークのクライアント計算機からのファイルオブジェクトのアクセス要求を処理するための第1の転送プロセスを複数個、同時並列的に起動させ、前記キャッシュ更新手段は、各々が1つの前記ファイルオブジェクトの取得を行なうための第2の転送プロセスを複数個、同時並列的に起動させ、前記第1の転送プロ

セスの数を検知して、前記第1の転送プロセスの数と前記第2の転送プロセスの数との和が予め定められる上限以下となるように前記第2の転送プロセスの新たな起動を制御する、請求項1記載のゲートウェイ装置。

【請求項4】 前記キャッシュ更新手段は、前記ファイル転送記録手段の中から、キャッシュにヒットしたファイルオブジェクトの転送記録を抽出し、当該ファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納する、請求項1記載のゲートウェイ装置。

【請求項5】 前記キャッシュ更新手段はさらに、前記ファイル転送記録手段の末尾から転送記録を順次読出して格納し、先に格納された転送記録から順に出力するFIFOバッファ手段を含み、

前記キャッシュ更新手段は、前記FIFOバッファ手段から転送記録を読出して、当該転送記録のファイルオブジェクトを前記第2のネットワークのサーバ計算機から取得して前記キャッシュファイル手段に格納する、請求項1記載のゲートウェイ装置。

【請求項6】 前記キャッシュ更新手段は、前記ファイル転送記録手段から抽出した転送記録のファイルオブジェクトの前の更新が所定時間以内であれば、前記第2のネットワークのサーバ計算機へのアクセスを行なわない、請求項1記載のゲートウェイ装置。

【請求項7】 クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、

前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、当該演算結果に基づいて前記第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含むゲートウェイ装置。

【請求項8】 クライアント計算機とゲートウェイ装置とが第1のネットワークで接続され、サーバ計算機とキャッシュサーバ計算機とが第2のネットワークで接続され、複数の前記ゲートウェイ装置と複数の前記キャッシュサーバ計算機とが第3のネットワークで接続される分散ファイルシステムであって、

前記ゲートウェイ装置は、前記第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、

当該演算結果に基づいて前記第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力

するキャッシュサーバ計算機を選択するための中継制御手段とを含む分散ファイルシステム。

【請求項 9】 クライアント計算機が接続される第 1 のネットワークと、サーバ計算機が接続される第 2 のネットワークとの間に介在するように設けられるゲートウェイ装置であって、

前記第 1 のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一次的に定まる所定の演算を行ない、該演算結果に基づいてゲートウェイ装置を選択し、該選択が他のゲートウェイ装置であれば、当該アクセス要求を第 3 のネットワークを介して前記他のゲートウェイ装置へ出力するための中継制御手段と、

前記第 2 のネットワークのサーバ計算機から取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、

過去の、一定時間内に行なわれたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、前記中継制御手段が自身のゲートウェイ装置を選択した場合と前記第 3 のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に回答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在しない場合には、当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含むゲートウェイ装置。

【請求項 10】 サーバ計算機と複数のキャッシュサーバ計算機とが第 2 のネットワークで接続され、前記複数のキャッシュサーバ計算機と第 1 のネットワークによって接続されるクライアント計算機であって、キャッシュサーバ計算機に対するファイルオブジェクトのアクセス要求を出力し、当該ファイルオブジェクトを取得するためのファイル取得手段と、キャッシュサーバ計算機へのファイルオブジェクトに対するアクセス要求を出力する際、当該アクセス要求に含まれる所定のデータに演算結果が一次的に定まる所定の演算を行ない、当該演算結果に基づいて前記複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含むクライアント計算機。

【請求項 11】 前記中継制御手段はさらに、第 3 のネットワークを介して他のクライアント計算機からのファ

イルオブジェクトに対するアクセス要求を受信するための受信手段を含み、

前記中継制御手段は、前記他のクライアント計算機からのアクセス要求に含まれる所定のデータに演算結果が一次的に定まる所定の演算を行ない、当該演算結果に基づいて前記複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択する、請求項 10 記載のクライアント計算機。

【請求項 12】 クライアント計算機と第 1 のネットワークによって接続され、かつサーバ計算機と第 2 のネットワークによって接続される複数のキャッシュサーバ計算機と第 3 のネットワークによって接続されるゲートウェイ装置であって、

前記第 1 のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一次的に定まる所定の演算を行ない、該演算結果に基づいてゲートウェイ装置を選択し、該選択が他のゲートウェイ装置であれば、当該アクセス要求を第 3 のネットワークを介して前記他のゲートウェイ装置へ出力するための中継制御手段と、

前記第 2 のネットワークのサーバ計算機から取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、

過去の、一定時間内に行なわれたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、前記中継制御手段が自身のゲートウェイ装置を選択した場合と前記第 3 のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に回答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトが前記キャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含むゲートウェイ装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワーク上に分散した複数のサーバ計算機と、複数のクライアント計算機とがゲートウェイ装置（計算機）を介して通信回線で相互に接続されている分散ファイルシステムに関し、特に、複数のクライアント計算機がゲートウェイ装置を介してサーバ計算機内部の記憶装置内のファイルオブジェクトにアクセスする場合の分散ファイルシステムのフ

ファイル中継を行なうゲートウェイ装置、ゲートウェイ装置にアクセス要求を出力するクライアント計算機、およびそれらを接続した分散ファイルシステムに関する。

#### 【0002】

【従来の技術】以下の説明で「サーバ計算機」とは、ネットワーク上において何らかのサービスを提供している計算機をいい、「クライアント計算機」とは、そうしたサーバ計算機のサービスを要求し、受ける計算機をいうものとする。また「ファイルオブジェクト」とは、分散ファイルシステムの利用するネットワークプロトコルと、ネットワークアドレス（サーバ計算機の名称）と、ファイル名称と、ファイルの実体との組をいうものとする。ここで「ファイルオブジェクトの名称」とは、ネットワークプロトコルとネットワークアドレスとファイル名称との組のことをいう。

【0003】従来、上述したような分散ファイルシステムにおいては、複数のクライアント計算機からのサーバ計算機上のファイルオブジェクトに対する読出要求は、一旦途中のゲートウェイ計算機で中継されていた。ゲートウェイ計算機がサーバ計算機から読出したファイルオブジェクトは、クライアント計算機に中継される。それと同時にそれらファイルオブジェクトは、ゲートウェイ計算機内部のキャッシュファイル（固定ディスク、半導体メモリなど）に蓄積される。そうしたゲートウェイ計算機の一例が特開平4-313126号公報（発明の名称「分散ファイルシステムのファイル入出力方式」）に開示されている。

【0004】図15は、この従来の分散ファイルシステムを示している。このシステムは、ゲートウェイ計算機110と、このゲートウェイ計算機110を介して相互に接続されたサーバ計算機100および複数のクライアント計算機120を含む。サーバ計算機100は、複数のクライアント計算機120によって共有されている。

【0005】サーバ計算機100は、ファイルを格納するディスク装置136と、ネットワークを介してディスク装置136のファイルの内容を入出力するためのファイル入出力応答手段134と、ディスク装置136内のブロック情報をネットワークを介してクライアント計算機120に応答するためのブロック情報応答手段132とを含む。

【0006】クライアント計算機120は、サーバ計算機100のブロック情報応答手段132に対してブロック情報要求を送り、ブロック情報を受けるためのブロック情報要求手段152と、ゲートウェイ計算機110を介してサーバ計算機100のファイル入出力応答手段134に対してファイル入出力要求を送り、当該ファイルを受信するためのファイル入出力要求手段154とを含む。

【0007】ゲートウェイ計算機110は、クライアント計算機120（複数）とサーバ計算機100との間に

介在し、サーバ計算機100内のディスク装置136と、各クライアント計算機120との間のキャッシュメモリとして機能するものであり、キャッシュした内容を格納するためのディスクキャッシュ144と、ファイル入出力要求手段154およびファイル入出力応答手段134の間に介在するキャッシュ管理手段142とを含む。

【0008】このシステムは次のように動作する。まず、クライアント計算機120がサーバ計算機100内のディスク装置136内のファイルにアクセスして1つのブロックを読み込む場合の動作について説明する。ブロック情報要求手段152は、ゲートウェイ計算機110を介してサーバ計算機100に対して読込対象のブロックにかかるブロック情報の取得を要求するブロック情報要求メッセージを発行する。

【0009】サーバ計算機100内のブロック情報応答手段132は、このメッセージに回答して、ディスク装置136から該当するブロック情報を取出し、そのブロック情報を有するブロック情報応答メッセージをゲートウェイ計算機110を介してクライアント計算機120内のブロック情報要求手段152に返却する。

【0010】クライアント計算機120のファイル入出力要求手段154は、返却されたブロック情報に基づいて読込対象のブロックの読込を要求するファイルアクセス要求を、ゲートウェイ計算機110内のキャッシュ管理手段142に対して発行する。

【0011】このファイルアクセス要求を受取ったキャッシュ管理手段142は、そのファイルアクセス要求にかかるブロック情報とディスクキャッシュ144に記憶されている読込対象のブロックにかかるブロック情報との比較を行なう。そしてディスクキャッシュ144に、該当するブロック情報が存在しない場合、または両方のブロック情報の内容（更新時間）が異なる場合（読込対象のブロックのキャッシュが有効でない場合）には、キャッシュ管理手段142は上述のファイルアクセス要求をサーバ計算機100のファイル入出力応答手段134に対して発行する。

【0012】サーバ計算機100のファイル入出力応答手段134は、このファイルアクセス要求に回答して、ディスク装置136内のファイルをアクセスして、該当するブロックを読み込み、ゲートウェイ計算機110にそのブロックを転送する。

【0013】キャッシュ管理手段142は、このブロックをディスクキャッシュ144に格納し、そのブロックにかかるブロック情報のディスクキャッシュ144への設定または更新を行なう。キャッシュ管理手段142は同時に、そのブロックをクライアント計算機120のファイル入出力要求手段154に対して転送する。

【0014】以上でディスクキャッシュ144内に有効なファイルが存在しない場合のこのシステムの動作につ

いて説明した。

【0015】次回に同一ファイルに対するアクセス要求があった場合には、ゲートウェイ計算機110のキャッシュ管理手段142は、ディスクキャッシュ144から当該ブロックを取出し、アクセス要求を発行したクライアント計算機120のファイル入出力要求手段154に対して当該ブロックを転送する。したがってこの場合、ゲートウェイ計算機110からサーバ計算機100に対する転送要求は発行されない。したがって中継アクセスの高速化が図られる。

【0016】類似の技術が、特開昭63-200244号公報（発明の名称「ファイル・アクセス・システム」）や、特開昭63-20184号公報（発明の名称「遠隔ファイル・アクセス・システム」）にも開示されている。いずれの技術においても、あるファイルオブジェクトに対するアクセス要求が発行された場合に、当該ファイルオブジェクトの内容が更新されているかどうかを確認するために、サーバ計算機の保持するファイルオブジェクトの変更時刻をキャッシュファイルの変更時刻と比較する。そしてキャッシュファイルの内容が古い場合には、クライアント計算機は更新されたサーバ計算機の内容を読出すようになっている。しかし、これら2件の公報に記載された技術では、キャッシュファイルはクライアント計算機内に存在しており、特開平4-313126号公報に開示されたシステムのようにシステム内にゲートウェイ計算機が含まれているわけではない。

【0017】しかし、これら3件の従来技術文献は、サーバ計算機のファイルオブジェクトと、キャッシュされたファイルオブジェクトとの内容が食い違わないような工夫をしているという共通点を持っている。

【0018】このように本来同一であるべきファイルの食い違いを防ぐことをインテグリティ（Integrity）を保つという。また、食い違いの起きたファイルのデータをステールデータ（stale data）と呼ぶ。

【0019】上述した従来の技術の基本は、ネットワーク経由でサーバ計算機の、該当するファイルオブジェクトの変更時刻を調べ、対応するキャッシュファイルの内容の変更時刻と比較するアルゴリズムを使用している。そのため、サーバ計算機に対するネットワーク経由の問合せを必ず行なわなければならない。ネットワークの実効的な転送速度が小さい場合や、該当するファイルオブジェクトを有するサーバ計算機にかかっている負荷が高くて即座に応答できない場合には、そうした問合せ自体にかなり時間がかかってしまうことがある。そのためネットワークファイルシステムによっては、このようなファイルキャッシュのシステムを採用していない場合もある。すなわちこうしたシステムでは、サーバ計算機の保持するファイルオブジェクトが更新されているにもかかわらず、ゲートウェイ計算機の持つキャッシュ内の対応するファイルオブジェクトは必ずしも更新されてい

い。このようなネットワークファイルシステム方式の代表的なものとして、いわゆるインターネットにおけるHTTP（Hyper Text Transfer Protocol）を使った広域分散型マルチメディア情報システムがある。

【0020】インターネットは、その基本プロトコルとしてTCP/IPプロトコルを利用した、グローバルなネットワークである。インターネット上に構築された地域分散型マルチメディア情報提供システムは、World Wide Web（WWW）と呼ばれる。WWWは、ネットワーク上に分散したファイルオブジェクトを扱うことができる。これらのファイルオブジェクトは、単なるテキストデータにとどまらず、画像データ、音声データ、ビデオ画像データなどさまざまな種類のものを含む。WWWは、情報提供側にとっても、情報利用者（ユーザ）にとっても魅力的であるため、ネットワーク上のWWWに関するトラフィックが爆発的に増加している。WWWシステムでは、クライアント計算機のユーザは、グラフィカルユーザインタフェースを持ったブラウザソフトウェアをクライアント計算機上で実行させるだけで、ネットワーク上に分散したサーバ計算機の保持するさまざまなファイルオブジェクトで構成された情報を次々とアクセスすることができる。WWWは、このような操作の簡単さのために近時めざましい普及をみせている。

【0021】そしてこのWWWシステムは、TCP/IPプロトコルの上に構築されたHTTPプロトコルでファイルオブジェクトを転送している。

【0022】HTTPプロトコルを実施する上においては、特開平4-313126号公報に記載されたようなゲートウェイ計算機によりファイルオブジェクトを中継転送することが広く行なわれている。データは、ゲートウェイ装置にキャッシュされる。またWWWシステムでは、ファイルオブジェクトの中継転送においてステールデータが発生することは容認されている。すなわちゲートウェイ計算機にキャッシュされている内容が、サーバ計算機の当該ファイルオブジェクトと一致していない場合を許容している。

【0023】キャッシュファイル内のステールデータの率をステールデータ率と呼ぶこととする。何らかの方式でキャッシュファイルの内容を更新しない限り、ステールデータ率は増加する。一般にキャッシュファイルのステールデータ率が増加すると、クライアント計算機を操作しているユーザは、内容が古いと自主的に判断して、ファイルオブジェクトをサーバ計算機から再度ロードすることを試みる。この場合、キャッシュファイルの内容を利用しないプロトコルが用いられる。この結果、ゲートウェイ計算機内のキャッシュファイルシステムが意味を持たなくなる傾向がある。

【0024】一方で、このようなステールデータの発生を容認するプロトコルの利点もある。クライアント計算機が、サーバ計算機のファイルオブジェクトのアクセス

をゲートウェイ計算機に中継依頼してアクセスを行なった場合、ゲートウェイ計算機のキャッシュファイルにヒットした場合には、キャッシュファイルからファイルオブジェクトが読出されてクライアント計算機に転送される。サーバ装置に対する当該ファイルオブジェクトの変更時刻の間合せは不要である。そのため短時間にファイルオブジェクトがキャッシュから取出され、クライアントに返送される。

【0025】HTTPプロトコルのようにステールデータを容認するネットワークファイルプロトコルは、全地球的な広域ネットワークで利用される上で有利な点を有している。すなわち、ゲートウェイ装置のキャッシュに当該ファイルオブジェクトがキャッシュされていれば、サーバ装置に対するアクセスが発生しない。その結果応答時間短縮を図ることができるという有利な点を持っている。

【0026】上述のようにネットワーク上のファイルオブジェクトのアクセスを高速化し、かつインターネットのトラフィックを低減させるゲートウェイ計算機は特に「proxyサーバ装置」とも呼ばれている。

【0027】「proxy」とは「代理人」という意味である。proxyサーバ装置は、その名のとおり、クライアント計算機がネットワーク上のサーバ計算機のファイルオブジェクトにアクセスする場合に、そのアクセス要求をサーバ計算機を代理して受付け、ファイルオブジェクトの転送を中継する機能を果たす。proxyサーバ装置の概念を図16を参照して説明する。

【0028】proxyサーバ装置13は、たとえばある企業内に用いられた内部ネットワーク23と、その企業外の外部ネットワーク（インターネットなど）25との間に介在して設けられるものである。内部ネットワーク23は複数のクライアント計算機24を含み、外部ネットワーク25は、同じく複数のサーバ計算機11を含んでいる。各サーバ計算機11は、ファイルオブジェクト12を保有している。

【0029】proxyサーバ装置13は、キャッシュファイル16と、クライアント計算機からリード要求19を受け、キャッシュファイル16内に当該ファイルオブジェクトが存在する場合にはそのファイルオブジェクトをデータ20として返送し、キャッシュファイル16内に当該ファイルオブジェクトの有効なものが存在しない場合には外部ネットワーク25を介してサーバ計算機11に対するリード要求17を発行し、対応するファイルオブジェクトをデータ18として受けてクライアント計算機24に対して返送するproxyプロセス14と、proxyプロセス14によるファイルオブジェクトの転送記録を記録するアクセスログ15とを含む。proxyプロセス14は、サーバ計算機11からファイルオブジェクトを取得したときには、当該ファイルオブジェクトをキャッシュファイル16にも新たに書込む機

能を有している。

【0030】このproxyサーバ装置13は次のように動作する。まず、内部ネットワーク23内のクライアント計算機24が、外部ネットワーク25上のサーバ計算機11のファイルオブジェクト12に対するリード要求19をproxyサーバ装置13内のproxyプロセス14に対して発行する。proxyプロセス14はこのリード要求19に応答して、キャッシュファイル16をアクセスし、キャッシュファイル中に当該ファイルオブジェクトのキャッシュデータがあるか否かを判定する。当該ファイルオブジェクトがキャッシュされていれば、proxyサーバ装置13固有のキャッシュファイル有効期限と、キャッシュされたファイルオブジェクトの最終変更時刻とを比較する。キャッシュが有効期限内であれば、キャッシュファイルからファイルオブジェクト読出（22）、そのデータ20を内部ネットワーク23のリード要求19を発行したクライアント計算機24に対して返送する。

【0031】キャッシュファイル16内に該当ファイルオブジェクトが存在しない場合、および存在していたとしても有効期限が過ぎている場合には、キャッシュファイル16内に有効なファイルオブジェクトが存在しないものと判定される。その場合、proxyプロセス14は、外部ネットワーク25のサーバ計算機11に対して、ファイルオブジェクト12の読出要求17を発行する。これに応答してサーバ計算機11は、ファイルオブジェクト12をデータ18としてproxyプロセス14に返送する。proxyプロセス14は、ログファイル15にアクセスログとして、ファイルオブジェクト名称（すなわちネットワークプロトコル名称と、サーバ計算機のネットワークアドレス名称と、ファイル名称との組）と、ファイルオブジェクトのデータの実体（すなわちファイルオブジェクトそのもの）と、書込時刻とを書込む（26）。

【0032】proxyプロセス14はさらに、内部ネットワーク23内のクライアント計算機24にデータ20を転送し、かつキャッシュファイル16に、当該ファイルオブジェクトを書込む（21）。

【0033】したがってキャッシュファイル16内には、ファイルオブジェクト名称と、ファイルオブジェクトの実体と、ファイルオブジェクトをサーバ装置から取得した時刻（最終変更時刻）とが記録されている。

【0034】proxyサーバ装置は、ヨーロッパ原子核物理研究所CERNで開発されたhttpdや、日本の電子技術総合研究所の佐藤豊氏の開発されたDelete Gateなどのソフトウェアを、ネットワーク接続されたUNIXの動くコンピュータ上で実行することにより実現されることが一般的である。

【0035】proxyサーバ装置の物理構成を図17に示す。proxyサーバ装置は、UNIXの動作する

ワークステーション200により構成される。ワークステーション200は、CPU（中央処理装置）202と、CPU202に対して内部バス204により接続されるメモリ206と、ファイル用のI/O（入出力）装置208と、ルータ210を介して外部ネットワークに接続される第1ネットワークI/Oインタフェース212と、内部ネットワークに接続される第2ネットワークI/Oインタフェース214を含む。I/O装置208には、キャッシュファイルの蓄積、アクセスログ、および各種ワークファイルの記憶場所として使用されるファイル部216が接続されている。

【0036】このワークステーション200は、CPU202が、上述したhttpdやDeleGateというソフトウェアを実行することによりproxyサーバ装置13を実現する。

【0037】図18は、proxyサーバ装置の他の構成例を示す図である。このproxyサーバ装置はワークステーション300からなっている。ワークステーション300は、概略、図17のワークステーション200と同様の構成であるが、図17の第2ネットワークI/Oインタフェース214に代えて、モデム装置304に接続されるシリアルポート302を有している点が異なっている。そしてこのワークステーション300は、モデム装置304および公衆電話回線網306を介して内部ネットワークに接続されている。

【0038】図18において、図17と同一の部品には同一の参照番号を付している。それらの名称および機能も同一である。したがって、ここではそれらについての詳しい説明は繰返さない。

【0039】図19にproxyサーバ装置のさらに他の構成例を示す。図19のproxyサーバ装置も、同じワークステーション400により実現される。このワークステーション400は、概略、図17に示されるワークステーション200と同様の構成であるが、第1ネットワークI/Oインタフェース212および第2ネットワークI/Oインタフェース214に代えて、内部ネットワーク23に接続された1つのネットワークI/Oインタフェース402を有している点で異なっている。内部ネットワーク23は、ルータ210を介して外部ネットワーク25に接続されており、また複数のクライアント計算機24にも接続されている。ワークステーション400は、内部ネットワーク23およびルータ210を介して外部ネットワーク25のサーバ計算機11と通信するとともに、内部ネットワーク23およびクライアント計算機24を介してユーザ31と通信する\*

$$Vave(kbps) = 64kbps \times (1 - Hitrate) + 10000kbps \times Hitrate$$

【0048】キャッシュファイルが存在しない場合、Hitrateは0である。したがって平均速度(kbps)は、数1から外部ネットワークのアクセス速度64kbpsと等しくなる。また、Hitrateが1であ

\*とができる。

【0040】図19と図17とにおいて、同一の部品には同一の参照符号および名称を与えている。それらの機能も同一である。したがって、ここではそれらについての詳しい説明は繰返さない。

【0041】proxyサーバ装置は、既に述べたように中継データのキャッシュ機構を有する。すなわち、proxyサーバ装置はその内部に存在するキャッシュファイル装置に、中継するファイルオブジェクトのデータをキャッシュする。

【0042】proxyサーバ装置のキャッシュファイルには、proxyサーバ装置固有の有効期限が定められている。ファイルオブジェクトがキャッシュに書込まれてから有効期限内に、同じサーバ計算機上のファイルオブジェクトへのアクセス要求をクライアント計算機から受取った場合には、キャッシュファイルに以前に書込まれたファイルオブジェクトを取出してクライアント計算機に転送する。これによりサーバ計算機に対するアクセスは発生しない。

【0043】このようなゲートウェイ装置によるキャッシュの効果を以下具体的に説明する。proxyサーバ装置はある会社組織に設置し、外部のネットワークとは64kbpsの速度で接続するものとする。また、このproxyサーバ装置は一方で、10Mbpsの速度の会社組織内のローカルエリアネットワークに接続されるものとする。外部と内部のネットワークの速度差は1桁以上ある。このような例はよく見られる例であり、その典型的な例が図19に示す構成である。

【0044】このとき、proxyサーバ装置で中継ファイルオブジェクトをキャッシュするものとして、キャッシュのヒット率をHitrateとする。Hitrateは、ファイルオブジェクトに対するアクセス要求に対してキャッシュが100%ヒットするとき1であり、0%ヒットするとき0になる指数である。

【0045】なお、proxyサーバ装置におけるキャッシュファイルからのデータ読出およびキャッシュファイルへのデータ書込速度は、ネットワーク速度に比べて十分高速であるものとする。すなわち、キャッシュファイルに対するファイルオブジェクトの入出力に要する時間は無視できるものとする。

【0046】このとき、外部ネットワークにあるファイルオブジェクトを内部からアクセスするときの平均アクセス速度Vaveは次の数1によって表わされる。

【0047】

【数1】

れば、すなわち100%キャッシュファイルにヒットするのであれば、平均速度(kbps)は、内部ネットワークの速度と等しく10,000kbps(=10Mbps)となる。つまり、ヒット率が高いほどクライアン



ト計算機から見た平均アクセス速度は高くなる。ヒット率と平均アクセス速度との関係を示す数1をグラフ化したものを図20に示す。

【0049】ところで、キャッシュヒット率は、キャッシュファイルの量に依存している。すなわち、ファイルオブジェクトをなるべくたくさんキャッシュファイルの中に蓄積しておけば、再びアクセスされたファイルオブジェクトやキャッシュファイル中に含まれる確率は高く、ヒット率は高くなる。そうした関係を示すものとして、キャッシュヒット率と、キャッシュファイルの蓄積

量との関係を図21にグラフとして示す。図21に示すように、キャッシュファイルサイズを小さくすればキャッシュヒット率は0に近く、キャッシュファイルサイズを大きくすればキャッシュヒット率は徐々に上昇する。

【0050】proxyプロセスにおいては、キャッシュファイルの内容を維持する方式として、最初のファイルアクセスによる書込が起きてから一定期間経過したファイルオブジェクトを無効としていく制御方式が取られている。この一定期間を有効期限と呼んでいる。この有効期限は、proxyサーバ装置を管理する者が実情に

合わせて適切な値を設定する。有効期限を長く取れば、キャッシュファイルに蓄積されるデータ量は増加する。その結果、キャッシュのヒット率は高くなる。すなわち、キャッシュのヒット率を上げるには、キャッシュの有効期限を長く取ればよい。

【0051】なお、キャッシュファイルを格納するファイル装置の記憶容量は有限であるから、有効期限を無制限に長くするわけにはいかない。有効期限が過ぎたキャッシュファイルは、何らかのタイミングで消去される。このようにキャッシュファイルを消去するまでの時間

【0052】

【教2】

キャッシュ有効期限  $\propto$  キャッシュファイル蓄積量  
 $\propto$  キャッシュヒット率

【0054】この関係から、図22に示したように、キャッシュ有効期限を増やすとキャッシュのヒット率が高まることがわかる。

【0055】キャッシュ有効期限とファイルオブジェクトの量との関係は、クライアント計算機を利用する内部ユーザが、外部ネットワーク上のサーバ計算機のファイルオブジェクトをアクセスする頻度によって異なる。キャッシュ有効期限とキャッシュファイル蓄積量とは、ほ

ぼ比例するものの、その比例係数はネットワーク速度、内部クライアント計算機ユーザ数などの使用環境により変動する。

【0056】なお、キャッシュファイル1日程度の有効期限では、キャッシュヒット率は12%程度である。有効期限を14日にするとキャッシュヒット率は41%程度になることが経験されている。有効期限14日のときにはファイルオブジェクトが700MB程度キャッシュファイルに蓄積されていることが観察されている。

【0057】また、近年ローカルネットワークの規模が大きくなり、接続されるクライアント計算機の数が増大するにつれ、またはクライアント計算機の読出要求が増大するにつれて、単一のproxyサーバ計算機では、各クライアント計算機からの要求に十分に応えることが難しくなっている。そこで、複数のproxyサーバ計算機を組合せて、ファイルオブジェクトに対するアクセス処理の効率の向上を図る方式が提案されており実施されている。

【0058】図23は、複数のproxyサーバ計算機を用いたシステムの構成例である。proxyサーバ計算機501は、広域ネットワークとローカルネットワーク間の中継を行ない、ファイルオブジェクトをキャッシュする。このproxyサーバ計算機501は、以後2次キャッシュproxyサーバ計算機と呼ぶ。これに対して、proxyサーバ計算機503～508は、1次キャッシュproxyサーバ計算機と呼ばれ、さらに分割されたローカルネットワークに接続されるクライアント計算機24と2次キャッシュproxyサーバ計算機501間の中継およびファイルオブジェクトのキャッシュを行なう。

【0059】たとえば、クライアント計算機24から1次キャッシュproxyサーバ計算機504に対してファイルオブジェクトに対する読出要求を出力したとする。このとき、1次キャッシュproxyサーバ計算機504のキャッシュファイル510内に有効なファイルオブジェクトが存在する場合、1次キャッシュproxyサーバ計算機504は、キャッシュファイル510よりファイルオブジェクトを取出し、読出要求を出したクライアント計算機24に送信する。

【0060】また、1次キャッシュproxyサーバ計算機504に有効なファイルオブジェクトが存在しなかった場合、クライアント計算機24からの読出要求は2次キャッシュproxyサーバ計算機501に中継される。2次キャッシュproxyサーバ計算機501は、同様にキャッシュファイル502内の有効なファイルオブジェクトの有無をチェックし、当該ファイルオブジェクトが存在すれば1次キャッシュproxyサーバ計算機504に送出し、存在しなければ読出要求を広域ネットワーク61を介してサーバ計算機11に中継する。

【0061】以降の処理は、単体のproxyサーバ計



算機と同様であるので、ここではその詳しい説明は繰返さない。

【0062】この方式によって、各々の1次キャッシュproxyサーバ計算機503～508は、クライアント計算機を分担して受け持つことができ、負荷の分散が図れる。また、2次キャッシュproxyサーバ計算機501も、各1次キャッシュproxyサーバ計算機503～508でキャッシュされていなかったファイルオブジェクトに関してのみ処理を行なえばよいので、負荷\*

$$(1 - (\text{一次キャッシュproxyサーバ計算機のキャッシュヒット率})) \times N \text{ 台}$$

【0064】しかしながら、この方式では1台の2次キャッシュproxyサーバ計算機501が広域ネットワーク61に接続されるサーバ計算機11との中継をすべて行っており、本質的な負荷の分散が達成されているとは言い難い。

【0065】また、キャッシュファイルの使用にも無駄が多い。すなわち、1次キャッシュproxyサーバ計算機群503～508の有するファイルオブジェクトの内容はすべて2次キャッシュproxyサーバ計算機501も保持していると考えられる。これは、中継経路上にある2次キャッシュproxyサーバ計算機501と1次キャッシュproxyサーバ計算機503～508では必ずファイルオブジェクトがキャッシュされるからである。また、各1次キャッシュproxyサーバ計算機同士でも、別の1次キャッシュproxyサーバ計算機に接続されるクライアント計算機が同一のファイルオブジェクトを讀出している場合、同一のファイルオブジェクトを重複して持っている可能性が高い。

【0066】したがって、分散ファイルシステム全体でのキャッシュ容量は2次キャッシュproxyサーバ計算機501のキャッシュファイル502の容量であるといえるので、1台の2次キャッシュproxyサーバ計算機だけではキャッシュファイルの容量には限界があるといえる。

【0067】このような問題を解決するために、図24のような方式が提案され実施されている。

【0068】この方式では、図23の2次キャッシュproxyサーバ計算機501に相当するproxyサーバ計算機は存在しない。各proxyサーバ計算機601～606は、クライアント計算機群を分担して受け持っている。たとえば、proxyサーバ計算機602に接続されたクライアント計算機24からの読出要求は、proxyサーバ計算機602に受信される。ここで、proxyサーバ計算機602のキャッシュファイル608に有効なファイルオブジェクトが存在する場合、当該ファイルオブジェクトを取出し、読出要求を出したクライアント計算機24に送出する。

【0069】proxyサーバ計算機602のキャッシュファイル608に有効なファイルオブジェクトが存在

\*の軽減が図れる。具体的には、全体でN台のクライアント計算機24が存在する場合、従来の方式ではproxyサーバ計算機は、N台のクライアント計算機の要求を処理する必要があったが、この方式では数3で算出される台数分のクライアント計算機からの要求を処理すればよいことになる。

【0063】

【数3】

しなかった場合、proxyサーバ計算機602はローカルネットワーク600に接続される他のproxyサーバ計算機601、603～606に対し、それぞれに接続されるキャッシュファイル607、609～612内に有効なファイルオブジェクトを保持しているかどうかを問合せる。有効なファイルオブジェクトを保持しているproxyサーバ計算機が存在する場合には、たとえば、proxyサーバ計算機605がキャッシュファイル611に有効なファイルオブジェクトを保持している場合、proxyサーバ計算機605はproxyサーバ計算機602に対してファイルオブジェクトを送信し、proxyサーバ計算機602は当該ファイルオブジェクトを讀出要求を出したクライアント計算機24に送出する。

【0070】ローカルネットワーク600に接続されたproxyサーバ計算機601、603～606のキャッシュファイル607、609～612に有効なファイルオブジェクトが存在しなかった場合、proxyサーバ計算機602は広域ネットワーク61を経由して、サーバ計算機11にクライアント計算機24からの読出要求を中継する。

【0071】以降の処理は、単体のproxyサーバ計算機と同様であるので、ここでの詳細な説明は繰返さない。

【0072】この方式により、同一のファイルオブジェクトを複数のproxyサーバ計算機が保持することがなくなり、キャッシュファイルの有効利用が図れる。また、分散ファイルシステム全体でのキャッシュ容量は、全proxyサーバ計算機601～606のキャッシュファイル607～612の容量の合計となり、キャッシュの大容量化が図れる。したがって、高いキャッシュヒット率が期待でき、実行転送速度の向上が期待できる。

【0073】また、図25のような方式も提案され実施されている。図23の方式との違いは、2次キャッシュproxyサーバ計算機705～707が複数用意され、1次キャッシュproxyサーバ計算機711～716が2次キャッシュproxyサーバ計算機705～707にクライアント計算機群からの読出要求を中継する場合、その読出要求の要求先であるサーバ計算機70

1, 702のアドレス名称によって、たとえば、名称がxxx.eduならばeduドメインサーバ計算機702に対する読出要求と判断して、proxyサーバ計算機705にアクセス要求を出力する。また、名称がyy.y.comならばcomドメインサーバ計算機702に対する読出要求と判断してproxyサーバ計算機706にアクセス要求を出力する。このように、末尾のドメイン名称(edu, com等)による簡単なルールでアクセス要求を中継する2次キャッシュproxyサーバ計算機705~707を選択している。

【0074】その他の処理は単一のproxyサーバ計算機と同一であるので、ここでの詳細な説明は繰返さない。

【0075】これにより、同一の2次キャッシュproxyサーバ計算機に読出要求が集中することがなくなり、負荷の分散が図れる。この方式においても、同一のファイルオブジェクトを複数の2次キャッシュproxyサーバ計算機が保持する必要はなく、キャッシュファイルの有効利用が図れる。また、分散ファイルシステム全体でのキャッシュ容量は全2次キャッシュproxyサーバ計算機のキャッシュファイルの容量の合計となり、キャッシュの大容量化が図れる。したがって、高いキャッシュヒット率が期待でき、実行転送速度の向上が期待できる。

#### 【0076】

【発明が解決しようとする課題】上述したproxyサーバ装置においては、キャッシュ有効期限をある程度の長さに設定することによりキャッシュファイルに蓄積されるファイルオブジェクトの容量が増え、キャッシュヒット率が高まり効率的なファイルオブジェクトの転送が可能になる。

【0077】しかし、一方、サーバ計算機のファイルオブジェクトの内容は、修正変更されることがあるため、キャッシュファイルの有効期限が長いと、その期間内はサーバ計算機内のファイルオブジェクトの内容が変更されているにもかかわらず、内部ネットワークのクライアント計算機のユーザはキャッシュファイルに蓄積された古い内容のファイルオブジェクトを読出すことになる。

【0078】たとえば、天気図や電子新聞等のファイルオブジェクトは、同一のファイルオブジェクト名称でありながら数時間おきに情報更新されるものが多い。したがって、たとえば、キャッシュ有効期限を24時間とすると、3時間おきに更新される天気図、電子新聞などの情報は7/8以上の確率で古くて役に立たない情報となってしまう。

【0079】図26は、このような従来のproxyサーバ装置におけるキャッシュ制御のデータの流れをタイムチャートにしたものである。A, B, C, D, E等の名称は同一ファイルオブジェクト名称でありながら内容が異なるデータであることを示している。クライアント

計算機がproxyサーバ装置経由でアクセス(ユーザアクセス(1))すると、サーバ計算機のファイルオブジェクトの内容Bがproxyサーバ計算機によってキャッシュされ、クライアント計算機にはBが転送される。次に、クライアント計算機からユーザアクセス

(2)が送信されると、proxyサーバ装置にキャッシュされている内容Bは24時間有効なので、キャッシュヒットしてファイルオブジェクトの内容Bがクライアント計算機に転送される。しかしこのときには、既にサーバ計算機内では内容がEとなっているのに、クライアント計算機はBという内容の古いデータを見ていることになる。

【0080】このようなキャッシュファイルの内容のうち、外部ネットワークに接続されたサーバ計算機内に存在する元のデータが変更されている割合をステールデータ率(stale data rate)と呼ぶ。したがって、キャッシュ有効期限を増やすとキャッシュファイルに蓄積されるファイルオブジェクトの容量が増加してキャッシュヒット率が高くなるが、キャッシュファイル内の内容のステールデータ率が増加することになる。

【0081】上述したCERNのhttpdソフトウェアで構成されるproxyサーバ装置であれば、特定のサーバ計算機のファイルオブジェクトのキャッシュ有効期限を小さくできるが、この方法ではクライアント計算機がファイルオブジェクトにアクセスを行なうと、キャッシュ有効期限が短いためキャッシュにミスヒットしてからサーバ計算機にファイルオブジェクトに対するアクセスを行なうことになる。したがって、クライアント計算機のユーザは、proxyサーバ計算機によるキャッシュの恩恵を受けられず、広域ネットワークによる低速なファイルオブジェクト転送が終了するまで待たされることになる。

【0082】したがって上述した従来の技術では、キャッシュファイルの中のファイルオブジェクトの内容の新鮮さとキャッシュヒット率の維持はそれぞれ相反する問題となる。

【0083】一般に、キャッシュファイルのステールデータ率が増加すると、クライアント計算機のユーザは内容が古いはずであると自主的に判断してファイルオブジェクトをサーバ計算機から直接ロードするようになる。この場合キャッシュファイルを通り越して、実際にサーバ計算機からロードすることを指定するプロトコルが利用される。その結果、proxyサーバ計算機のキャッシュファイルが意味を持たなくなることになる。

【0084】また、従来よりproxyサーバ計算機の負荷を分散する方法として、上述したように図23~図25の方式が提案されている。しかしいずれの場合にも十分に負荷が分散されているとはいえない。図23に示す方式では、1台の2次キャッシュproxyサーバ計算機501が広域ネットワークに接続されているサーバ

計算機との中継をすべて行なっているので、負荷の分散が達成されているとはいえない。

【0085】図24の方式では、proxyサーバ計算機601～606と広域ネットワーク61との間の経路が多重化されていない。また、ローカルネットワーク600に接続されるproxyサーバ計算機601～606は、この方式に対応した特殊なプロトコルを理解するものに置き換える必要がある。

【0086】図25の方式では、1次キャッシュproxyサーバ計算機711～716は、ドメイン名等の単純なルールで分散を行なうので、2次キャッシュproxyサーバ計算機705～707間での負荷の偏りが生じる。

【0087】本発明は、上記問題点を解決するためになされたもので、請求項1に記載の発明の目的は、キャッシュファイルの内容を新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0088】請求項2に記載の発明の目的は、キャッシュファイルの内容のうちサーバ計算機における更新頻度の高いファイルオブジェクトを新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0089】請求項3に記載の発明の目的は、通常の利用に悪影響を与えることなく、キャッシュファイルの内容を新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0090】請求項4から6に記載の発明の目的は、キャッシュファイルの内容を新鮮に保つとともに、キャッシュヒット率も向上させることである。

【0091】請求項7または8に記載の発明の目的は、ファイルオブジェクトに対するアクセス要求を分散して経路を有効に利用し、スループットを向上させることである。

【0092】請求項9から11に記載の発明の目的は、ファイルオブジェクトに対するアクセス要求を分散して経路を有効に利用し、スループットを向上させ、さらにローカルネットワーク上の通信量を削減することである。

【0093】請求項12に記載の発明の目的は、ファイルオブジェクトに対するアクセス要求を分散して経路を有効に利用し、スループットを向上させることである。

【0094】

【課題を解決するための手段】請求項1に記載の発明は、クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するように設けられるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求にตอบสนองして、当該ファイルオブジェクトにより指定される第2のネットワークのサーバ計算機に対して、当該ファイルオブジェクトに対するアクセス要求を行なうネットワーク

中継手段を含み、ネットワークファイル中継手段は、取得したファイルオブジェクトを所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、過去の一定期間内に行なわれたファイルオブジェクトの転送記録を蓄積するファイル転送記録手段と、ファイルオブジェクトに対するアクセス要求にตอบสนองして当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含み、ゲートウェイ装置はさらに、ファイル転送記録手段から所定の規則に従って転送記録を抽出し、当該転送記録のファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納するためのキャッシュ更新手段を含む。

【0095】ネットワークファイル中継手段によって転送されたファイルオブジェクトは、所定の規則に従ってキャッシュ更新手段により更新される。キャッシュ更新手段は、有効期限が切れたものはもちろん、有効期限内のファイルオブジェクトも第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。そのため、キャッシュの有効期限を長めに取っていても、キャッシュの有効期限よりも短い間隔で変更されるサーバ計算機のファイルオブジェクトがキャッシュに反映される確率は高くなる。

【0096】請求項2に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はさらに特定の文字列パターンを予め記憶するための文字列パターン記憶手段を含み、キャッシュ更新手段は特定の文字列パターンと一致するファイルオブジェクト名称を含む転送記録をファイル転送記録手段から抽出し、当該ファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。

【0097】キャッシュ更新手段は、特定の文字列パターンと一致するファイルオブジェクトのみ更新アクセスを行なうので、予めファイルオブジェクトの更新頻度の高いファイルオブジェクト名称を登録しておけば不要なキャッシュ更新アクセスを削減でき、キャッシュファイルの内容を新鮮に保つとともにキャッシュヒット率も向上させることができる。

【0098】請求項3に記載のゲートウェイは、請求項1に記載のゲートウェイ装置であって、ネットワークファイル中継手段は、各々が第1のネットワークのクライアント計算機からのファイルオブジェクトのアクセス要

求を処理するための第1の転送プロセスを複数個、同時並列的に起動させ、キャッシュ更新手段は各々が1つのファイルオブジェクトの取得を行なうための第2の転送プロセスを複数個、同時並列的に起動させ、かつ第1の転送プロセスの数を検知して、第1の転送プロセスの数と第2の転送プロセスの数との和が予め定められる上限以下となるように第2の転送プロセスの新たな起動を制御する。

【0099】第1のプロセスが多数発生しているときは第2のプロセスの起動を抑制し、第1のプロセスが僅かしか発生していないときには第2のプロセスを多数起動させることができるので、通常のユーザによるファイルオブジェクトのアクセス要求が、キャッシュ更新手段による悪影響を受けることがない。また、通常のユーザによるアクセス要求が少ないときには随時更新処理が行なえるので、キャッシュの新鮮さを保つことができる。

【0100】請求項4に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はファイル転送記録手段の中から、キャッシュファイル手段からキャッシュヒットしたファイルオブジェクトの転送記録を抽出し、当該ファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。

【0101】ユーザによるアクセス要求がキャッシュにヒットした場合、キャッシュ更新手段はその直後にサーバ計算機へファイルオブジェクトのアクセス要求を行なう。したがって別ユーザが同一ファイルオブジェクトをアクセスした場合、最新のファイルオブジェクトが既にキャッシュに格納されていることになる。

【0102】請求項5に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はさらにファイル転送記録手段の末尾から転送記録を順次読出して格納し、先に格納された転送記録から順に出力するFIFOバッファ手段を含み、キャッシュ更新手段はFIFOバッファ手段から転送記録を讀出して、当該転送記録のファイルオブジェクトを第2のネットワークのサーバ計算機から取得してキャッシュファイル手段に格納する。

【0103】FIFOバッファ手段は、ファイル転送記録手段の末尾から転送記録を順次格納し出力するので、キャッシュ更新手段の処理が容易になる。

【0104】請求項6に記載のゲートウェイ装置は、請求項1に記載のゲートウェイ装置であって、キャッシュ更新手段はファイル転送記録手段から抽出した転送記録のファイルオブジェクトの前の更新が所定時間以内であれば、第2のネットワークのサーバ計算機へのアクセスは行なわない。

【0105】キャッシュ更新手段は、所定時間以内に再び同じファイルオブジェクトに対するアクセス要求があった場合に後のアクセス要求に対するキャッシュ更新処

理を行なわない。このことによって、不要なキャッシュ更新アクセスを削除し、無駄なトラフィックを抑制して回線を有効に使うことができる。

【0106】請求項7に記載の発明は、クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、当該演算結果に基づいて第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含む。

【0107】中継制御手段は、複数のキャッシュサーバ計算機の中からファイルオブジェクトに対するアクセス要求を出力するキャッシュサーバ計算機を選択する場合に、演算結果が一意的に定まる演算を行なうことによってキャッシュサーバ計算機を一意的に定める。したがって同一のファイルオブジェクトに対するアクセス要求は同一の経路によって転送され、異なるファイルオブジェクトに対するアクセス要求は分散して経路を転送することになるので、経路が有効に利用され、スループットが向上することになる。

【0108】請求項8に記載の発明は、クライアント計算機とゲートウェイ装置とが第1のネットワークで接続され、サーバ計算機とキャッシュサーバ計算機とが第2のネットワークで接続され、複数のゲートウェイ装置と複数のキャッシュサーバ計算機とが第3のネットワークで接続される分散ファイルシステムであって、ゲートウェイ装置は、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行なうための演算手段と、当該演算結果に基づいて第3のネットワークの複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含む。

【0109】各ゲートウェイ装置は、アクセス要求に含まれる所定のデータに同一の演算を行なうため複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択する際、同一のファイルオブジェクトであれば同一のキャッシュサーバ計算機を選択することになる。したがって同一のファイルオブジェクトが複数のキャッシュサーバ計算機内に含まれるということがなくなり、キャッシュの有効利用が可能となる。

【0110】請求項9に記載の発明は、クライアント計算機が接続される第1のネットワークと、サーバ計算機が接続される第2のネットワークとの間に介在するよう

に設けられるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、この演算結果に基づいてゲートウェイ装置を選択し、この選択が他のゲートウェイ装置であれば、アクセス要求を第3のネットワークを介して他のゲートウェイ装置へ出力するための中継制御手段と、第2のネットワークのサーバ計算機から取得したファイルオブジェクトを所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、過去の一定時間内に行われたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、中継制御手段が自身のゲートウェイ装置を選択した場合と第3のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に回答して、当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在する場合は当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより所定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含む。

【0111】このゲートウェイ装置は、複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択する機能と、ファイルオブジェクトをキャッシュする機能とを有することによって、ローカルネットワーク上で30の通信量を削減することを可能にしている。

【0112】請求項10に記載の発明は、サーバ計算機と複数のキャッシュサーバ計算機とが第2のネットワークで接続され、複数のキャッシュサーバ計算機と第1のネットワークによって接続されるクライアント計算機であって、キャッシュサーバ計算機に対するファイルオブジェクトのアクセス要求を出力し、当該ファイルオブジェクトを取得するためのファイル取得手段と、キャッシュサーバ計算機へのファイルオブジェクトに対するアクセス要求を出力する際、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいて複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択するための中継制御手段とを含む。40

【0113】クライアント計算機に、複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択する機能を含ませることによって、ローカルネットワーク上で通信量を削減し、クライアント計算機の有効利用を図っている。

【0114】請求項11に記載のクライアント計算機 50

は、請求項10に記載のクライアント計算機であって、中継制御手段はさらに、第3のネットワークを介して他のクライアント計算機からのファイルオブジェクトに対するアクセス要求を受信するための受信手段を含み、中継制御手段は他のクライアント計算機からのアクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいて複数のキャッシュサーバ計算機の中からアクセス要求を出力するキャッシュサーバ計算機を選択する。

【0115】このクライアント計算機は、他のクライアント計算機からのアクセス要求を受信して、当該アクセス要求に含まれる所定のデータに基づいて複数のキャッシュサーバ計算機の中からキャッシュサーバ計算機を選択している。したがって同じローカルネットワークに接続されるクライアント計算機の中の一部にキャッシュサーバ計算機を選択する機能を持たせて、クライアント計算機の有効利用を図っている。

【0116】請求項12に記載の発明は、クライアント計算機と第1のネットワークによって接続され、かつサーバ計算機と第2のネットワークによって接続される複数のキャッシュサーバ計算機と第3のネットワークによって接続されるゲートウェイ装置であって、第1のネットワークのクライアント計算機からのファイルオブジェクトに対するアクセス要求に対して、当該アクセス要求に含まれる所定のデータに演算結果が一意的に定まる所定の演算を行ない、当該演算結果に基づいてゲートウェイ装置を選択し、この選択が他のゲートウェイ装置であれば、当該アクセス要求を第3のネットワークを介して他のゲートウェイ装置へ出力するための中継制御手段と、第2のネットワークのサーバ計算機から取得したファイルオブジェクトを、所定の有効期限が経過するまで一時的に蓄積するとともに、最終変更時刻をファイルオブジェクト単位で記録するためのキャッシュファイル手段と、過去の一定時間内に行なわれたファイルオブジェクトの転送記録を蓄積するためのファイル転送記録手段と、中継制御手段が自身のゲートウェイ装置を選択した場合と第3のネットワークを介して他のゲートウェイ装置からアクセス要求を受信した場合に、ファイルオブジェクトに対するアクセス要求に回答して当該ファイルオブジェクトの最終変更時刻を参照し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在する場合には当該ファイルオブジェクトをクライアント計算機に転送し、有効期限内のファイルオブジェクトがキャッシュファイル手段に存在しない場合には当該ファイルオブジェクトにより指定されるサーバ計算機の当該ファイルオブジェクトを取得してクライアント計算機に転送するための転送手段とを含む。

【0117】ゲートウェイ装置に、キャッシュサーバ計算機を選択する機能と、ファイルオブジェクトをキャッシュする機能とを持たせることによって、ローカルネッ

トワーク上の通信量を削減し、キャッシュサーバ計算機の負荷を軽減することを可能にしている。

【0118】

【発明の実施の形態】

【実施の形態1】本発明が適用されるゲートウェイ装置(proxyサーバ装置と呼ぶ)は、ネットワークインタフェースを備えたUNIX OSの動作する計算機によって、以下の説明および図面を参照すれば容易に実現可能である。そうしたproxyサーバ装置の構成の一例は、図19に示したとおりである。図19のproxyサーバ装置については既に説明したので、その構成についての詳細な説明はここでは繰返さない。

【0119】通常は、ユーザ31がクライアント計算機24を使用してproxyサーバ装置(図19におけるproxyサーバ装置400)に対して、サーバ計算機11のファイルオブジェクト10を取得するようネットワーク23を経由して依頼を出す。ファイルオブジェクト12は、前述したようにプロトコル名称と、サーバ計算機のネットワークアドレスと、ファイルオブジェクトの名称と、ファイルデータの実体との組からなっている。

【0120】図1に本願発明に係るproxyサーバ装置の第1の実施の形態における装置53のシステム構成を示す。proxyサーバ装置53は、内部ネットワーク23および外部ネットワーク25に接続された第1のproxyプロセス14と、第1のproxyプロセス14が使用するキャッシュファイル16と、第1のproxyプロセス14が、アクセスログ(転送記録)としてファイルの名称を記録するアクセスログ15と、アクセスログ15の末尾から順次取出して格納し、先に格納されたアクセスログから順に出力するFIFOバッファ35と、予め作成されたアクセスパターンリストを格納するパターンファイル30と、FIFOバッファ35からアクセスログを取出してパターンファイル30に格納されたアクセスパターンと比較し一致する場合にはこのアクセスログをメモリバッファURLBUF(URLについては後述する)に格納するキャッシュ更新プロセス36と、URLBUFに格納されたアクセスログに含まれるファイルオブジェクト名称をもとに、当該ファイルオブジェクトをサーバ計算機11から取得してキャッシュファイル16に格納する第2のproxyプロセス34とを含む。

【0121】第1のproxyプロセス14と、第2のproxyプロセス34と、キャッシュ更新プロセス36とはいずれも、たとえば図19に示すメモリ206内にプログラムとして格納され、CPU202によってこれを実行することにより実現される。

【0122】第1のproxyプロセス14が、特許請求の範囲に記載の転送手段に相当し、キャッシュファイル16がキャッシュファイル手段に相当し、アクセスロ

グ15がファイル転送記録手段に相当する。また、キャッシュ更新プロセス36と、パターンファイル30と、FIFOバッファ35と、第2のproxyプロセス34とによりキャッシュ更新手段が構成される。

【0123】第2のproxyプロセス34は、本願発明に係るゲートウェイ装置に特有なものであるが、次のような条件に従って起動される。

【0124】(1) 第2のproxyプロセス34は、第1のproxyプロセス14と共通のキャッシュファイル16を使用する。

【0125】(2) 第2のproxyプロセス34のキャッシュ有効期限は0に設定される。すなわち、第2のproxyプロセスを経由するサーバ計算機11へのアクセスは、必ず外部ネットワーク25内のサーバ計算機11に対して行なわれ、ファイルオブジェクトを取得してキャッシュファイル16に格納する。

【0126】図1に示すproxyサーバ装置53の動作について以下説明する。内部ネットワーク23内のクライアント計算機24が、このproxyサーバ装置53に対してネットワークアクセス要求19を与えるものとする。第1のproxyプロセス14が、このネットワークアクセス要求19を受け取り、まずI/Oインタフェース202(図17、図18、図19参照)を経由してファイル装置216内に存在するキャッシュファイル16をアクセスする(21)。また第1のproxyプロセス14は、このネットワークアクセス要求とキャッシュファイル16内のファイルオブジェクト名称とを比較し(22)、一致しているか否かを判定する。一致したものがあれば第1のproxyプロセス14はさらに、当該ファイルオブジェクトのキャッシュファイル16内の最終変更時刻を抽出し(22)、現在の時刻と比較する。そして現在の時刻が、当該ファイルオブジェクトの最終変更時刻を起点とする有効期限内であれば、キャッシュファイル16から当該ファイルオブジェクトを抽出し(22)、ネットワークI/Oインタフェース402を経由してクライアント計算機24にファイルオブジェクトを返す(20)。

【0127】該当するファイルオブジェクトがキャッシュファイル16内に存在していない場合、または存在していてもそのキャッシュ有効期限が切れている場合には、第1のproxyプロセス14は、ネットワークI/Oインタフェース402を経由して外部ネットワーク25のサーバ計算機11にファイルオブジェクト転送要求17を送る。このときのサーバ計算機11は、当該ファイルオブジェクト内のネットワークアドレス名称により特定される。

【0128】該当するサーバ計算機11は、要求されたファイルオブジェクトを第1のproxyプロセス14に返送する(18)。すなわちこのファイルオブジェクトが第1のproxyプロセス14により取得される。

【0129】第1のproxyプロセス14は、このファイルオブジェクトをキャッシュファイル16にその最終変更時刻とともに書込み(21)、アクセスログ15にこのファイルオブジェクトの名称を記録する(26)。

【0130】一方、本願発明のproxyサーバ装置53は、キャッシュ更新プロセス36と第2のproxyプロセス34とをCPU202により処理走行させてファイルオブジェクトを更新する点において従来のproxyサーバ装置と相違する。

【0131】キャッシュ更新プロセス36および第2のproxyプロセス34は次のように動作する。第1のproxyプロセス14によってアクセスログファイル15に格納されたアクセスログの末尾からFIFOバッファ35に順に格納される(28)。FIFOバッファ35は、先に格納されたアクセスログから順に出力する(27)。キャッシュ更新プロセス36は、FIFOバッファ35からアクセスログを1行ずつ抽出し、メモリバッファに格納する。キャッシュ更新プロセス36は、メモリバッファからアクセスログを取出し、それぞれのアクセスログをもとにキャッシュにヒットしているか否かを判定する。キャッシュにヒットしていればさらに、パターンファイル30からアクセスパターンリストを抽出し(29)、アクセスログに含まれるファイルオブジェクト名称がアクセスパターンリストと一致しているものを抽出する。またキャッシュ更新プロセス36は、抽出したファイルオブジェクトのうちファイルオブジェクト名称が特定のパターンに一致するものは除外して、これをメモリバッファURLBUFに格納する。

【0132】第2のproxyプロセス34は、URLBUFが空でないならば、URLBUFに格納されたファイルオブジェクト名称をもとに(31)、サーバ計算機11にアクセスし(32)、当該ファイルオブジェクトを取得して(33)キャッシュファイル16に格納する(37)。これによりキャッシュファイル16内のファイルオブジェクトが最新の内容に維持される。

【0133】このように、第2のproxyプロセス34を起動すると、大きく次の2つの効果を得ることができる。

【0134】(1) 第1のproxyプロセス14では、キャッシュファイルの有効期限がある値に設定されており、ファイルオブジェクトのキャッシュ更新処理をこの第1のproxyプロセス14を用いて行なうとすると、キャッシュファイル16にヒットしてしまう可能性が高い。これでは、外部ネットワーク25のサーバ計算機11のファイルオブジェクトに対するアクセスが行なわれないので、キャッシュファイル16の更新が実現できない。

【0135】一方、第2のproxyプロセス36は、キャッシュ有効期限が0になっており、第2のproxy

yプロセス36に対してアクセス要求を出せば、このキャッシュファイルはミスヒットしたことになるので、外部ネットワーク25を介してサーバ計算機11に対するファイルオブジェクトのアクセスが行なわれ(32)、当該ファイルオブジェクトは第2のproxyプロセス34に転送される(33)ことによって、キャッシュファイル16中のファイルオブジェクトは最新の状態に更新される。

【0136】(2) 第1のproxyプロセス14は、アクセスログを残すが、このアクセスログをもとにキャッシュ更新アクセスをするべきファイルオブジェクトが抽出される。したがって、もしキャッシュ更新プロセス36が第1のproxyプロセス14を使用してサーバ計算機11のファイルオブジェクトのアクセスを行なうと、その結果として生成されるアクセスログがアクセスファイル15に記録されてしまう。キャッシュ更新プロセス36は、次の更新アクセス動作においてアクセスログファイル15に格納されたアクセスログをもとにアクセスすべきファイルオブジェクト名称を抽出する。したがって、再度同じファイルオブジェクトをサーバ計算機11から取得することになり、同一のファイルオブジェクトが多重に出現するループ動作が起きてしまうことになる。これを避けるために第2のproxyプロセス34を動作させ、第2のproxyプロセス34を利用したキャッシュ更新アクセス動作を行なう。

【0137】本発明を非常に有効に適用できる例として、インターネット上のWWWシステムがある。WWWシステム上でのproxyサーバ装置において、本願発明のキャッシュ更新プロセスを行なう手順を以下に説明する。

【0138】WWWシステムでは、ネットワーク上に分散したファイルオブジェクト名称はUniform Resource Locator(URL)と呼ばれる形式で表現され、特定される。URLの一例を次に示す。

【0139】

【数4】

<http://www.xxx.co.jp/test/index.html>

【0140】数4においては「http」は使用するプロトコルを示す。「www.xxx.co.jp」は、ネットワーク上のHTTPサーバ計算機のアドレスを示すものであり、ネットワーク上で同一のものはないよう選ばれている。また「/test/index.html」はサーバ計算機内のファイル名称を示す。

【0141】第1のproxyプロセス14として一般に利用されているDeleGateを使用する場合は、そのアクセスログの例は次のように、クライアント計算機名称、時刻、“HTTPプロトコル(ファイル取得要求)”、その他、およびキャッシュヒット状況(キャッシュヒットした場合H)などとなる。以下に、そのアクセスログの例を示す。



【0142】

\* \* 【数5】

クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/HTTP/1.0"その他H  
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/sharpcolor.gif HTTP/1.0"その他H  
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/isg.gif HTTP/1.0"その他H  
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/Zaurus.gif HTTP/1.0"その他H  
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/Shoin.gif HTTP/1.0"その他H  
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/Prostation.gif HTTP/1.0"その他H  
 クライアント計算機名称 [時刻] "GET http://naragw.sharp.co.jp/S2.gif HTTP/1.0"その他H

【0143】数5に示すアクセスログは、図2に示すような、出願人がインターネット上で公開している情報ページへアクセスした場合のアクセスログに相当する。図2に示されるページは、テキストと、6個のグラフィックデータとから構成されているので、このページに対するアクセスを1回行なうと、6個のグラフィックデータに対するアクセスを含む7つのアクセスログが形成される。すなわち、上述のアクセスログにおいて第1行目はこの情報ページのテキストに対するアクセス要求であり、残りの6行は表紙に貼り込まれた出願人会社のロゴのグラフィックデータ (sharpcolor.gif ※20

10※f)、出願人会社内の情報システム事業本部と呼ばれる事業部のロゴ (isg.gif)、出願人会社の4つの製品の写真のグラフィックデータ (Zaurus.gif, Shoin.gif, Prostation.gif, S2.gif) の取得要求を示している。

【0144】HTTPプロトコルは、コマンド文字列 (GET等) と、URLと、プロトコルバージョン (「HTTP/1.0」等) から構成されている。以下はその一例である。

【0145】

【数6】

"GET http://www.xxx.co.jp/test/index.html HTTP/1.0"

【0146】HTTPプロトコルについては、最後に掲げる表内に示された参考文献のうち、参考文献3および参考文献4に記載されている。キャッシュ更新プロセスの実施手順について以下に説明するが、そのための前提条件は次の(1)および(2)となっている。

【0147】(1) 第1のproxyプロセス14は、キャッシュ有効期限M(時間)で起動する。たとえばM=24時間とする。

【0148】(2) 第2のproxyプロセス34は、①キャッシュファイルを第1のproxyプロセス14のものと共有し、②キャッシュ有効期限を0に設定する、という2つの条件に従って起動される。

【0149】このような条件の下に、本願発明のキャッシュ更新プロセスを毎日一定時刻に起動し、一定時刻に終了する。毎日一定時刻にキャッシュ更新プロセスを自動起動するには、UNIX OSに備わっているタイマ機能(cron)を使用する。これは、UNIX OSに関連してよく知られている機能であるので、その詳細についてはここでは説明しない。なお、UNIX OS 40以外のOSでも、同様の機能が提供されていることが多い。

【0150】図3は、本願発明のキャッシュ更新プロセスの処理手順を示すフローチャートである。このプロセス★

★Sを実現するためのプログラムは、たとえば図19のメモリ206内に配置され、CPU202によって実行される。このプロセスとネットワークおよびファイル装置216との入出力には、それぞれネットワーク1/Oインタフェース402とI/Oインタフェース208とが使用される。

【0151】まず、ステップS1において第1のproxyプロセス14が、アクセスログファイル15に格納されるアクセスログの末尾を1行読出す。読出せない場合は、アクセスログが読めるまで待機する。UNIXでは、このようなアクセスログの末尾を呼出す機能は、OSとともに提供されているコマンドtailで可能であり、その標準出力から1行読出し、メモリ内に確保された文字列バッファであるLINEバッファに1行格納するには、以下のように行なう。ここで、第1のproxyプロセス14が格納したアクセスログが/usr/local/etc/telegated/logs/10001.httpであるとする。これは図1のアクセスログファイル15に格納されるアクセスログに相当する。

【0152】

【数7】

```
tail -f /usr/local/etc/delegated/logs/10001.http | read LINE
```

【0153】数7のtail -fによる1行読出は、図1の27に相当するが、これはソフトウェアによるFIFOバッファ35を構成しており、読出ポイントおよび書込ポイントをそれぞれ有している。

【0154】

【数8】

```

green [時刻] "GET http://naragw.sharp.co.jp/HTTP/1.0" その他H
green [時刻] "GET http://naragw.sharp.co.jp/sharpcolor.gifHTTP/1.0" その他H
rp--> green [時刻] "GET http://naragw.sharp.co.jp/isg.gifHTTP/1.0" その他H
green [時刻] "GET http://naragw.sharp.co.jp/Zaurus.gifHTTP/1.0" その他H
green [時刻] "GET http://naragw.sharp.co.jp/Shoin.gifHTTP/1.0" その他H
green [時刻] "GET http://naragw.sharp.co.jp/Prostation.gifHTTP/1.0" その他H
green [時刻] "GET http://naragw.sharp.co.jp/ZAURUS/index.htmlHTTP/1.0" その他H
wp--> green [時刻] "GET http://naragw.sharp.co.jp/S2.gifHTTP/1.0" その他H

```

【0155】すなわち、第1のproxyプロセス14が書込むアクセスログは、1個のファイルオブジェクト転送ごとに1行発生するが、このとき書込ポインタ(wp)を1行進める。また、ソフトウェアによるFIFOバッファ35がtail -fでアクセスログを1行読出せば、読出ポインタ(rp)を1行進める。第1のproxyプロセス14による書込速度(行/秒)と、キャッシュ更新プロセス36におけるtail -fによる1行読出速度(行/秒)が一致していなくてもよい。第1のproxyプロセス14によるアクセスログ書込速度が大きくなると、書込ポインタ(wp)は先に進むが、キャッシュ更新プロセス34によってこれを読出ポインタ(rp)が追いつけながらアクセスログが読出される。アクセスログ書込速度が小さいか、読出側のキャッシュ更新プロセス34の処理が進行すれば、やがて読出ポインタ(rp)が書込ポインタ(wp)に追いつくことになる。これが、ソフトウェアFIFOバッファとしての動作である。

【0156】次に、LINEバッファに格納された1行のアクセスログ情報は過去にアクセスされたファイルオブジェクトの記録であるので、以下のような条件でキャッシュ更新アクセスを行なうべきファイルオブジェクト名称を含むものを選択する。この選択処理を、以後フィルタリングと呼び、図3に示すフローチャートのステップS2に相当する。このフィルタリングのルールは以下のとおりである。

【0157】(1) HTTPコマンド文字列がファイルオブジェクトに対するリード要求(GET)であるもの。

【0158】(2) ファイルオブジェクト名称のプロトコル部分がhttpであるもの。

(3) キャッシュにヒットしたアクセスであるもの。

【0159】(4) ファイルオブジェクト名称がキャッシュ更新アクセスに適さないものを除外する。

【0160】(5) 特定の文字列パターンを含むアクセスリストとファイルオブジェクト名称が一致するもの。

【0161】以上の(1)～(5)の条件を用いてフィルタリングを行ない、ファイルオブジェクト名称のフィールドだけを取り出して、キャッシュ更新アクセスでファ

イルオブジェクト名称を抽出する。

【0162】ファイルオブジェクト名称のプロトコル部分がhttpのもののみ抽出するのは、第1のproxyプロセス14および第2のproxyプロセス34として一般的に利用されているDeleGateソフトウェアがキャッシュファイルを生成するプロトコルはhttpプロトコルのみであるので、そのファイルオブジェクトのみはキャッシュ更新アクセスする意味があるからである。それ以外のプロトコルによるアクセスはファイルオブジェクトがキャッシュされないので、キャッシュ更新アクセスをする意味がない。

【0163】キャッシュにヒットしたもののみを抽出するのは、キャッシュにヒットしていないファイルオブジェクトはサーバ計算機11へのアクセスが行なわれたものと解釈されるので、その直後にキャッシュ更新アクセスを行なう必要はないと判断されるからである。

【0164】ファイルオブジェクト名称が、キャッシュ更新アクセスに適さないものとしては、URL中に「?」を含むものなどがある。「?」を含むURL表記は、サーバ計算機11に対してクライアント計算機24を使用するユーザが文字列を手入力する場合などに使われるため、この文字を含むURL表記を有するファイルオブジェクトをキャッシュ更新アクセスの対象とすると、予期せぬ結果を招くおそれがある。

【0165】特定の文字列パターンを含むアクセスリストと一致するもののみを選ぶのは、サーバ計算機において頻繁に内容更新が行なわれるものである天気図、電子新聞等を提供しているサーバ計算機など特定のものであることが多い。そこで、そのようなサーバ計算機のファイルオブジェクトを指定するような文字列パターンを格納したアクセスリストと一致したものだけをキャッシュ更新アクセスの対象とすることによって、不要なキャッシュ更新アクセスを極力削減するためのものである。

【0166】このような文字列パターンを格納したパターンファイル30を、ここでpattern.txtとし、その内容を数9に示すものとする。

【0167】

【数9】

weather  
news  
http://www.asahi.com  
http://www.sjmercury.com  
http://www.cnn.com  
http://naragw.sharp.co.jp/ZAURUS/  
daily

\*【0168】UNIXでは、このようなファイルオブジェクト名称の抽出作業は、OSとともに提供されているコマンド群である数10を用いて行なうことができる。

【0169】

【数10】

\*

/usr/bin/fgrep によるストリング・パターン抽出(パターンファイル指定可能)  
/usr/bin/grep によるストリング・パターンの抽出  
/usr/bin/grep -v によるストリング・パターンの否定抽出  
/usr/bin/awk による特定のフィールド・切りだし

【0170】たとえば、次のようなコマンドを入力することにより、ステップS2の処理を経たものがメモリ上の文字列バッファURLBUFに出力される。 ※

※【0171】

【数11】

URLBUF='echo \$LINE|grep -v "\?"|grep "\GET"|grep 'H\$'  
awk '{print \$7}'|fgrep -f pattern.txt'

【0172】なお、上述したフィルタリングルールに反していれば、文字列バッファURLBUFは空になる。この間の経過を以下に説明する。第1のproxyプロセス14により作成されたアクセスログファイル15の★

★形式は既に述べたように次のとおりである。

【0173】

【数12】

クライアント計算機名称—[21/Aug/1995:09:27:43+0900] "GET http://naragw.sharp.co.jp/S2.gifHTTP/1.0" その他 キャッシュヒット状況

【0174】キャッシュヒット時には“H”という文字がキャッシュヒット状況として行の末尾に出力される。 30

【0175】まず、grep -v "\?"に入力することで、?を含まない行を抽出し標準出力に出力する。それをさらにgrep "\GET"でその行がファイルオブジェクト取得要求であるGETコマンドである行のみ抽出し、標準出力に出力する。さらにgrep 'H\$'に入力することで行末が「H」で終了する、すなわちキャッシュにヒットしたものを選び標準出力に出力する。ここで、\$は行の末尾を表現する正規表現である。

【0176】さらに空白がフィールド間の区切であると考え、7番目のフィールドがファイルオブジェクト名称に相当するので、awk '{print \$7}'を使ってファイルオブジェクト名称のみを抽出している。 40

【0177】さらに、ファイルオブジェクト名称部分が、予め用意されたパターンファイルpattern.txtの中のいずれかの行に一致するか否かをfgrep -f pattern.txtによりマッチングを行なってその結果を文字列バッファURLBUFに出力する。

【0178】ステップS3では、文字列バッファURLBUFが空でないか否かを判定する。もし文字列バッ 50

ファURLBUFが空の場合には、ステップS1に戻り、空でない場合にはステップS4へ進む。

【0179】ステップS4では、第2のproxyプロセス34によって、キャッシュ更新プロセス36で抽出されたファイルオブジェクトをサーバ計算機11から取得する。文字列バッファURLBUFに格納されたキャッシュ更新アクセスの対象であるファイルオブジェクト名称に基づいて、第2のproxyプロセス34はサーバ計算機11へアクセスするための子プロセスを起動する。子プロセスをキャッシュ更新子プロセスと呼ぶ。キャッシュ更新子プロセスは、第2のproxyプロセス34を通じてネットワーク接続をサーバ計算機11に対して行なう。これにより、サーバ計算機11に格納されるファイルオブジェクトへのアクセスが発生し、キャッシュファイル16の内容が更新される。この処理の詳細は後述する。

【0180】ステップS5では、現在の時刻を調べ、指定時刻が過ぎていればプロセスを終了する。指定時刻が過ぎていなければステップS1に戻り、以上の処理を繰返す。

【0181】以上が、本願発明の第1の実施の形態におけるproxyサーバ装置におけるアルゴリズムであ

る。図7は、proxyサーバ計算機53のキャッシュ有効期限が24時間で、サーバ計算機11のファイルオブジェクトが同一名称でありながら、内容が3時間おきに更新される場合のファイルオブジェクトのデータがキャッシュされるようすを示している。

【0182】まず、ユーザアクセス(1)が発生したときに、キャッシュファイル16に該当するファイルオブジェクトがなかったとすると、サーバ計算機11からデータBがアクセスされ、キャッシュファイル16に書込まれるとともに、クライアント計算機24に転送される。

【0183】次に、ユーザアクセス(2)が発生すると、キャッシュにヒットしたデータBがクライアント計算機24に転送される。このとき、サーバ計算機11はデータDになっているので、クライアント計算機24は3時間程度古いステールデータを読んだことになる。しかし、この動作の結果アクセスログファイル15にファイルオブジェクト名称とキャッシュにヒットしたという情報を含むアクセスログが記録されるので、図3を用いて説明した動作に従い、proxyサーバ装置53が自発的にサーバ装置11にアクセスし、データEを読み出しキャッシュファイル16を更新する。

【0184】さらに、ユーザアクセス(3)が起きると、キャッシュされたデータEがクライアント計算機24に転送される。この動作をきっかけにしてアクセスログファイル15にファイルオブジェクト名称とキャッシュにヒットしたという情報を含むアクセスログが記録されるので、図3を用いて説明した動作に従い、proxyサーバ装置53が自発的にサーバ計算機11にアクセスし、データIを読み出しキャッシュファイル16を更新する。

```
lynx -source http://www.xxx.co.jp/test/index.html > temp_file
```

【0191】さらに、proxyプロセス経由のアクセスも、UNIX OSの環境変数http\_proxyを設定することにより、lynxから利用可能である。また、proxyプロセスがDeleGateのようなproxyソフトウェアであれば、URL表記にproxyを指定することによりproxy経由のアクセスが可能である。これについては参考文献12を参照されたい。

【0192】後者の方式を採用する場合について以下に説明する。proxyプロセスが、ネットワークアドレ※

```
lynx -source http://proxyserver:10001/_ - http://www.xxx.co.jp/
test/index.html > temp_file
```

【0194】したがって、本願発明を実施する際には、同様にしてproxyプロセスを第2のproxyプロセス34のポート番号に設定すればよい。

```
lynx -source http://proxyserver:10001/_ - $URLBUF > temp_file
```

【0196】本願発明では、このようなファイルオブジェクト(URL)のキャッシュ更新アクセスにより、第

※し、データIを読み出しキャッシュファイル16を更新する。

【0185】したがって、ユーザアクセス(4)が起きると、キャッシュされたデータIがクライアント計算機24に転送される。

【0186】以上の動作からわかるように、クライアント計算機24はステールデータを読んでいることになりすが、図26の場合に比べると最新に近い情報を読んでいることになる。ユーザによるアクセスの間隔が3時間以内であれば、ステールデータは発生する確率が非常に低くなり、サーバ計算機11の最新の情報を得ることが可能になることが理解できる。

【0187】なお、図3のステップS4のネットワークアクセス部の実現方法としては、キャッシュ更新プロセスから直接HTTPプロトコルを発生させればよいが、UNIXでは広く一般に利用されているlynx等のWWWクライアントプログラムを利用すれば、HTTPを発生するキャッシュ更新プロセスを新たに製作しなくてもHTTPに従ったネットワークアクセスを行なうキャッシュ更新プロセス操作を実現できる。その方法を以下に説明する。

【0188】lynxはUNIX OSの上で動作するプログラムであるので、以下の説明ではUNIXの記述を使用する。

【0189】lynxでは次のコマンドを指定すれば、指定URLを読み出してtemp\_fileというファイルに書込むことができる。

【0190】

【数13】

※スがproxyserverであるゲートウェイ計算機のTCP/IPの10001番のポート番号を利用しているプロセスであるならば、このproxyプロセス経由でサーバ計算機11のファイルオブジェクトhttp://www.xxx.co.jp/test/index.htmlにアクセスするためには次の数14のコマンドを指定すればよい。

【0193】

【数14】

★【0195】

【数15】

1のproxyプロセス14の管理するキャッシュファイル16を最新状態に保つことが目的である。したがって、temp\_file自体は利用しないので捨ててよい。OSがUNIXであれば、空ファイルとして/dev/nullが準備されているので、このファイルに書込むようにすれば実際にはファイルの生成は行なわないので効率的である。

【0197】次に、第1のproxyプロセス14および第2のproxyプロセス34のキャッシュ管理手法について説明する。proxyプロセスのキャッシュ管理手法は、参考文献5、参考文献12にあるような公知の技術であるキャッシュ制御機能を持ったDeleGateなどにおけるものである。DeleGateにおけるキャッシュ管理手法は、ソースコードの形で公開されているので以下にその手法を述べる。なお、本願発明は\*

lynx -source http://proxyserver:10001/\_-\$URLBUF > temp\_file

【0200】なお、第2のproxyプロセス34とキャッシュ更新プロセス36の通信はTCP/IP、たとえばUNIXのメッセージ通信で行なわれる。

【0201】たとえば、文字列バッファURLBUFにhttp://www.sharp.co.jp/sample/test.htmlという文字列が格納されているとする。キャッシュ更新プロセス36が数16のコマンドを実行すると、キャッシュ更新プロセス36はURL取得要求を第2のproxyプロセス34に出力する。第2のproxyプロセス34は、子プロセスを生成し、この子プロセスにURL取得の処理を任せて自身のプロセスは再びキャッシュ更新プロセス36からのURL出力要求を待つ。キャッシュ更新プロセス34による数16の実行によりURL取得を実行する第2のproxyプロセス34が生成した子プロセスは、図4のフローチャートに示す処理を行なう。

【0202】以下、第2のproxyプロセス34が生成した子プロセスの動作を図4のフローチャートを用い

/cache/プロトコル名称/サーバ計算機名称/ファイルオブジェクトの実体部分

【0207】したがって、http://www.sharp.co.jp/sample/test.htmlは数19に示すファイルとなる(S11)。

/cache/http/www.sharp.co.jp/sample/test.html

【0209】proxyプロセスはまず、数19に示すファイル名称に変換し、そのファイルが存在するかどうかOSのopen()のシステムコールにより既存のファイルがオープンできるかどうかにより調べる(S12)。ファイルが存在しなければ、ファイル新規作成モードで数19に示すファイル名のキャッシュファイルを作成する(S13)。また、キャッシュファイルに対して排他制御を行なって、他のプロセスが書き込めないよう

\*これと同様のキャッシュ管理手法が使われているキャッシュ機能付proxyサーバソフトウェアであれば、第1のproxyプロセス14および第2のproxyプロセス34として適用可能である。

【0198】上述したように、第2のproxyプロセス34がproxyserverという名称のシステムで動作しており、TCP/IPの10001番でURL取得要求を受付けるものとする。このとき、文字列バッファURLBUFに指定URL表記が格納されている場合に、キャッシュ更新プロセス36は、数16に記載のコマンドを実行することでtemp\_fileに指定URLのファイルオブジェクトが格納されると述べた。

【0199】

【数16】

て説明する。

【0203】proxyプロセスでは、キャッシュファイル16を持っているが、それは通常のUNIXファイルシステムの一部として作られる。キャッシュ用ディレクトリとして/cacheというパーティションを使用するとすると、数17に示すようにプロトコル名称がhttp、サーバ計算機名称www.sharp.co.jp、ファイルオブジェクトの実体部分がsample/test.htmlとなる。

【0204】

【数17】

http://www.sharp.co.jp/sample/test.html

【0205】このとき、キャッシュファイルは次の数18に示す名称規則でキャッシュファイルを生成しようとする。

【0206】

【数18】

※【0208】

【数19】

に排他ロックする(S14)。

【0210】次に、ネットワークを経由してURLで指定されたファイルオブジェクトをサーバ計算機11から取得する(S15)。取得したファイルオブジェクトをクライアント計算機24へ転送するとともにファイルオブジェクトを数19に示すファイル名称のキャッシュファイルに書込む(S16)。続いて、キャッシュファイルをクローズして排他ロックを解除する(S17)。

【0211】上記処理では、キャッシュファイルを排他ロックしているが、この排他制御はOSがUNIXであればシステムコールである`flock()`関数を使って排他ロックできる。排他制御を使用する理由は、`proxy`プロセスは1つのファイルオブジェクトの転送ごとに子プロセスを起動してその子プロセスにキャッシュ制御を行なわせるため、同じキャッシュファイルシステムに対して子プロセス間で競合が発生する場合があるからである。

【0212】なお、キャッシュファイルに書込んだとき、ファイル変更時刻`Tm`がファイル(数19)の属性として記録される。ファイル更新時刻`Tm`は、UNIX OSのファイルシステムを管理するOSがファイルの属性として必ず付加するものである。すなわち、キャッシュファイルは通常のOSのファイルシステムをそのまま使用しており、何ら特殊なファイルシステムではない。

【0213】次に、キャッシュファイルが既に存在した場合を説明する(S19～S26)。

【0214】キャッシュファイル/`cache/http/www.sharp.co.jp/sample/test.html`の最終変更時刻`Tm`をOSから読出す。最終変更時刻は、UNIXであれば、`fstat()`システムコールで得られるもので、過去から未来に向かって単調に増加する関数である(S19)。

【0215】現在時刻をシステムコールから読出し`Tnow`という変数に格納する(S20)。現在時刻`Tnow`は、OSのシステムコールから得られる。最終変更時刻`Tm`と、現在時刻`Tnow`を比較し、キャッシュ有効期限内かどうかを判定する(S21)。キャッシュ有効期限内であれば、キャッシュファイルを共有排他ロックして他のプロセスからの書込はできないが、読出はできるようにする(S22)。そして、既にあるキャッシュファイルを読出し、クライアント計算機24に転送する(S23)。さらに、キャッシュファイルをクローズし、共有排他ロックを解除する(S17)。

【0216】また、キャッシュ有効期限を過ぎている場合は、キャッシュファイルに対して排他制御を行なつて、他のプロセスが書込めないように排他ロックする

(S24)。次に、ネットワーク経由でURLで指定されたファイルオブジェクトをサーバ計算機11から取得する(S25)。取得したファイルオブジェクトをクライアント計算機24へ転送するとともに、ファイルオブジェクトをキャッシュファイル(数19)に書込む(S26)。そして、キャッシュファイルをクローズして排他ロックを解除する(S17)。

【0217】以上が、一般にソースコードが公開され、よく知られている`proxy`プロセスのキャッシュ制御手順であるが、本願発明ではこの性質を利用して、キャッシュファイル16を第1の`proxy`プロセス14と

第2の`proxy`プロセス34で共有している。第2の`proxy`プロセス34は、キャッシュ有効期限0であるから、図3のフローチャートのS4により、第2の`proxy`プロセス34経由のサーバ計算機11へのアクセスがキャッシュファイル16の更新が必ず行なえることがわかる。

【0218】第1の実施形態においては、クライアント計算機24の利用が始まる7時00分より起動し、クライアント計算機24の利用が少なくなる夜21時00分には終了するよう実施した。これにより、クライアント計算機24の利用のない時間帯には、キャッシュ更新プロセス36を排除し、当制御方式を採用した計算機のメモリ資源を無駄にしないようにしている。

【0219】以上説明したように、`proxy`サーバ計算機53はユーザが行なったファイルオブジェクトに対するアクセス要求がキャッシュヒットすれば、その直後にサーバ計算機11から当該ファイルオブジェクトを取得してキャッシュファイル16を更新するので、別のユーザが同一ファイルオブジェクトをアクセスした場合に、最新のファイルオブジェクトが既にキャッシュファイル16に格納されていることになる。したがって、キャッシュ有効期限を長めに取っていても、キャッシュの有効期限よりも短い間隔で変更されるサーバ計算機のファイルオブジェクトがキャッシュに反映される確率が高くなる。

【0220】また、同一ユーザが時間をおいてから同一ファイルオブジェクトをアクセスし直すと、キャッシュの内容が最新のものに更新されていることになる。また、特定のパターンに一致するファイルオブジェクトのみ更新アクセスの対象となるので、予めファイルオブジェクトの更新頻度の高いサーバ計算機を登録しておけば不要なキャッシュ更新アクセスを削減でき、無駄なトラフィックを抑制して回線を有効に利用することができる。

【0221】〔実施の形態2〕次に、本願発明のゲートウェイ装置の第2の実施形態を説明する。この実施形態2は、実施形態1と同様であるが、実施形態1に対応するフローチャート(図3)のステップS4を改良したものである。実施形態1では、ステップS4においてアクセスログファイル15からファイルオブジェクト名称を文字列格納バッファURLBUFに格納して、第2の`proxy`プロセス34を介してサーバ計算機11にアクセスしているが、第2の実施形態では処理の高速化を図るため複数のファイルオブジェクトに対するアクセスを、順不同で同時並行的に実行するものである。

【0222】本発明が実施できるTCP/IPのネットワークでは、通信はパケット単位で行なわれるため、外部ネットワークの複数のサーバ計算機との同時通信が見掛け上可能である。したがって、ステップS4におけるキャッシュ更新子プロセスを複数同時に実施することが

できる。

【0223】内部ネットワーク23には複数のクライアント計算機24が接続されるので、proxyサーバ装置53によるファイルオブジェクトの中継転送要求は、外部ネットワーク25上の異なる複数のサーバ計算機11へのアクセスが入り交じって行なわれる。したがって、ファイルオブジェクトに対するアクセスは外部ネットワーク25上のさまざまな経路を通じて行なわれるため、実際のファイルオブジェクトの転送速度は、経路の途中のボトルネックによって決まる。

【0224】そのうちの1つのボトルネックは、proxyサーバ装置53と外部ネットワーク25を結ぶ最初の通信路(図19における通信路127)である。さらには、外部ネットワーク25上への複数のサーバ計算機11への経路にはさまざまなボトルネックが存在する。このため、外部ネットワーク25からproxyサーバ計算機53への転送速度は、この通信路の最大転送速度より小さい転送速度となるのが常である。たとえば、proxyサーバ装置53と外部ネットワーク25を結ぶ通信路(図19における通信路127)の最大転送速度が64kbp/sであっても、海外にあるサーバ計算機からの転送速度はその10分の1以下であることがしばしばである。

【0225】したがって、proxyサーバ装置53と外部ネットワーク25を結ぶ最初の通信路127の最大転送容量に近い転送を行なうように、同時に複数のネットワーク接続を実施することで転送のスループットを上げることが可能になる。すなわちこのネットワーク接続を1個の子プロセスに対応させるとして、複数のキャッシュ更新子プロセスを起動し、この通信路127の転送容量が100%近く使用されるように、複数のネットワーク接続を並列に実施した方が処理が高速化される。

【0226】そこで、同時に複数のキャッシュ更新アクセスを実行するように、キャッシュ更新子プロセスの最大並列実行数をMAXPROCESS数と定める。これは、以下に説明するフローチャートではメモリ(図19\*

`lynx -source http://proxyserver:10001/_-$URLBUF > temp_file &`

【0231】最後に、指定の終了時刻になったかどうかを判定し、指定時刻を過ぎていればキャッシュ更新プロセスは処理を終了し、指定の終了時刻を過ぎていなければS1以下の処理を繰返す(S426)。

【0232】以上のような処理により、MAXPROCESS以下のプロセスで同時並列実行が可能になる。

【0233】この実施形態2により、キャッシュ更新処理速度を速め、常にキャッシュ更新処理を待つ状態にすることでキャッシュ更新処理の遅滞の拡大を防止することが可能となる。なお、パターンファイル30を用意して、特定ファイルオブジェクトだけを更新対象にしているS2の処理も効果を上げている。

\*のメモリ206)に置かれる定数である。この値をあまう大きく取ると、OSのプロセス手順などの管理資源(メモリ)を消費することになるので、システムの動作状況によって決める必要がある。この実施形態2では10とした。

【0227】以下、具体的な処理手順を図5を参照しながら説明する。まず、変数としてメモリ206中にprocesses変数を定義する。バックグラウンドで、現在走行しているキャッシュ更新子プロセスの数を求めてprocesses変数に代入する。プロセスの数はOSのプロセス管理テーブルから得られる(S421)。次にprocesses変数と指定された最大子プロセス数MAXPROCESSを比較する。もし、 $processes \geq MAXPROCESS$ ならばS423に移り、一定時間休止(sleep)する(たとえば10秒)。これは、先に走行しているキャッシュ更新子プロセスの処理を終了するのを待つために行なわれる(S423)。

【0228】次に、指定の終了時刻になったかどうかの判定を行なう。指定時刻を過ぎていれば、キャッシュ更新プロセスは処理を終了する。指定時刻を過ぎていなければ、S421以下の処理を繰返す(S424)。

【0229】また、S423で $processes < MAXPROCESS$ ならば、第2のproxyプロセス34にアクセス要求を出力することによって、第2のproxyプロセス34はキャッシュ更新子プロセスをバックグラウンドプロセスとして1つ起動する(S425)。UNIX OSにおいて、プロセスをバックグラウンドで実行させるには、コマンドラインに&の記号を付けて起動すればよい。文字列バッファURLBUFには、S2により取得されたファイルオブジェクト名称が格納されているので、数20に示すコマンドを実行すれば、キャッシュ更新子プロセスがバックグラウンドプロセスとして実施される。

【0230】

【数20】

【0234】以上説明したように、実施の形態2では、キャッシュ更新処理を同時に行なうことで、キャッシュ更新の開始から終了までが短縮されるので、第1のproxyプロセス14によるユーザからのアクセス処理の頻度に追従してキャッシュ更新処理を進めることが可能となる。

【0235】[実施の形態3] 実施の形態3においては、実施の形態2と同様に、キャッシュ更新子プロセスの並列実行を行なうが、さらにプロセスの制御を行なうことにより、内部ネットワーク23のクライアント計算機24を使用するユーザが第1のproxyプロセス14を使用して外部ネットワーク25のサーバ計算機11



に頻繁にアクセスしている時間帯に、キャッシュ更新プロセス36を実行しても、キャッシュ更新子プロセスの数が抑制される。すなわち、キャッシュ更新子プロセスには低い優先権が与えられることで、クライアント計算機24を使用するユーザのネットワーク利用を妨げないようにしたものである。

【0236】また、第1のproxyプロセス14を利用するユーザのネットワークアクセスは、通常1つのページが複数のファイルオブジェクトから構成されているため（たとえば、図2に示す情報ページ）、1つのページのアクセスにおいては、複数のファイルオブジェクトのアクセスが時間的に連続して発生する。したがって、キャッシュ更新アクセスの子プロセスの起動において、休止時間を設けて子プロセス起動を遅延させることにより、ユーザのページアクセスが終了してからキャッシュ更新アクセスが起きる確率を高めることができる。

【0237】また、同一のファイルオブジェクトに対するキャッシュ更新アクセスの条件が揃ったとき、すなわち、実施形態1および2のURLBUFが空でないとき（図3および図4のS3がyesの場合）一定時間の無効時間を設け、前回キャッシュが更新されてから所定時間以内であればサーバ計算機11に対するキャッシュ更新アクセスを行なわないことで、クライアント計算機24が頻繁にファイルオブジェクトアクセスを行なっているときに、キャッシュ更新アクセスでサーバ計算機11に対するアクセスが頻繁に発生することを防止するものである。

【0238】この処理の簡単な実現方法として、第2のproxyプロセス34のキャッシュ有効期限を0では\*

processes <= キャッシュ更新子プロセスの数+第1proxyのネットワークアクセス  
中継子プロセスの数

【0243】processes変数と指定された最大子プロセスMAXPROCESSを比較し、processes ≥ MAXPROCESSならばS433へ移り、一定時間休止(sleep)する（たとえば10秒）。これは、先に起動されたキャッシュ更新子プロセス数の処理が終了するのを待つものである。

【0244】次に、指定の終了時刻になったか否かを判定し、指定時刻を過ぎていればキャッシュ更新プロセスは処理を終了する。指定の終了時刻になっていなければ、S431に戻って処理を続行する（S434）。また、S432でprocesses < MAXPROCESSならば、抽出したファイルオブジェクト名称（URLBUFに格納されたもの）をもとに、第2のproxyプロセス36を用いてアクセスするキャッシュ更新子※

(sleep \$DELAYTIME; lynx -source http://proxyserver:10001/\_ - \$URLBUF  
> temp\_file)&

【0247】最後に、指定の終了時刻になったかどうか

\*なく、無効時間を設定して起動すればよい。この無効時間として、30分または1時間等とし、サーバ計算機11の変更頻度の平均の最小時間程度にすればよい。これにより、無効時間以内に実施形態1、2または3の方式でキャッシュ更新アクセスを行なっても、キャッシュにヒットすることになるのでサーバ計算機11にアクセスが発生しなくなる。

【0239】図6は、実施の形態3の処理を示すフローチャートである。図6は、実施形態2の処理を示すフローチャートである図5のS421とS425をS431とS435に置き換えたものである。

【0240】図6に示す処理を行なうに先立って、最大ネットワーク接続個数MAXPROCESSという定数を定める。この定数は、キャッシュ更新子プロセスの数と第1のproxyプロセス14がネットワークアクセスの中継のために起動した子プロセスの数を加算した値の最大を示す定数である。したがって、proxyサーバ装置53は、なるべくMAXPROCESS以下にサーバ計算機11へのネットワークアクセスを制限しようとする。図6のS1～S3の処理は、実施形態1、2と同様であるのでここでの説明は繰返さない。

【0241】バックグラウンドで走行しているキャッシュ更新子プロセスの数と、第1のproxyプロセス14のネットワークアクセス中継子プロセスの数を求めて加算してprocesses変数（数21）とする（S431）。

【0242】

【数21】

※プロセスをバックグラウンドプロセスとして1つ起動する。

【0245】このとき、子プロセスは数22に示すように、休止時間をDELAYTIME変数（単位は秒）としてOSの休止機能をsleepを使ってDELAYTIME秒間休止してからキャッシュ更新アクセスを行なうようにする。1ページWWWデータは、図2に示すように複数の画像ファイルオブジェクトからなる場合、キャッシュヒット時は10秒以下でproxyサーバ装置53からクライアント計算機24に転送可能であることが経験的に知られているので、DELAYTIMEは10秒程度としている。

【0246】

【数22】

を判定し、指定時刻を過ぎていればキャッシュ更新プロ

セスは処理を終了し、指定終了時刻を過ぎていなければ、S1へ戻り処理を繰返す(S436)。

【0248】以上のように、キャッシュ更新子プロセスの個数と第1のproxyプロセス14を利用しているユーザのネットワークアクセスの中継子プロセス数の合計を制限しながらキャッシュ更新子プロセスを制御する。この結果、第1のproxyプロセス14を利用しているユーザによるネットワークアクセスの数が増えると、キャッシュ更新子プロセスは起動されにくくなる。したがって、ユーザが第1のproxyプロセス14を利用している時間帯に、キャッシュ更新プロセス36を実行しても、キャッシュ更新子プロセスには低い優先権が与えられることで、ユーザのネットワーク活動を妨げないようにすることができる。

【0249】また、ユーザのファイルオブジェクトに対するアクセスがキャッシュにヒットした場合は、通常1ページの転送は、数秒以内に終了するが、キャッシュ更新子プロセスはDELAYTIME(秒)遅延してから起動されるので、DELAYTIMEを10秒程度に設定すればユーザのファイルオブジェクトアクセスとキャッシュ更新アクセスは時間的にずれて実施される(すなわちパイプライン)ので、proxyサーバ計算機53への負担を減らし、ユーザに快適なネットワーク利用を提供することが可能となる。

【0250】以上説明したように、実施の形態3ではユーザが第1のproxyプロセス14を利用している時間帯に、キャッシュ更新プロセス36を実行しても、キャッシュ更新子プロセスには低い優先権が与えられることで、ユーザのネットワーク利用を妨げない効果がある。また、遅延時間を導入することで、キャッシュ更新処理はユーザの連続したファイルオブジェクト転送処理が終了してから行なわれるので、proxyサーバ計算機53の処理負担を軽減し、ユーザへのファイルオブジェクト転送を最優先できる。また、無効時間が設定されるため、クライアント計算機24が無効時間以内に同一ファイルオブジェクトに対してアクセスを行ない、キャッシュ更新アクセスの条件が揃ってもキャッシュ更新アクセスはキャッシュにヒットすることになり、実質的にサーバ計算機11に対するアクセスが発生しない。これにより、クライアント計算機24が同一ファイルオブジェクトを頻繁にアクセスしても、時間のかかるサーバ計算機11へのキャッシュ更新アクセスが抑制されるのでサーバ計算機11への負担の軽減とキャッシュ更新処理の効率化が図れる。

【0251】〔実施の形態4〕実施の形態1～3は、proxyサーバ計算機に関するものであったが、実施の形態4ではファイルオブジェクトの中継制御のみを行なうゲートウェイ装置(redirector計算機)について説明する。図8は、キャッシュサーバ計算機(proxyサーバ計算機)65～67およびredire

ctor計算機72～77をローカルネットワーク71で接続する構成を示す図である。

【0252】redirector計算機72～77は、proxyサーバ計算機の1種でもあるが、実施の形態4のredirector計算機72～77は、proxyサーバ計算機65～67への中継のみを行ないキャッシュは行なわない。

【0253】以下、redirector計算機72～77の動作について詳細に説明する。

【0254】redirector計算機72～77は、複数のproxyサーバ計算機65～67とクライアント計算機群との間に介在し、proxyサーバ計算機65～67とはローカルネットワーク71で接続され、クライアント計算機群とはローカルネットワーク78で接続される。また、ファイルオブジェクトを提供するサーバ計算機11とproxyサーバ計算機65～67とは、広域ネットワーク61で接続される。

【0255】まず、クライアント計算機24からのファイルオブジェクトに対する読出要求をローカルネットワーク78を介してredirector計算機73内のredirectorプロセスが受信する。redirector計算機は、上述した図17～19に示すワークステーション200～400で実現可能である。すなわち、redirectorプロセスに相当するプログラムをメモリ206内に格納し、そのプログラムをCPU202が実行することによってこのredirector計算機の機能が実現される。redirectorプロセスは、受信した読出要求の内容に基づいてハッシュ関数の計算を行ないその計算結果に従って中継するproxyサーバ計算機65～67を選択する。そしてredirectorプロセスは、読出要求を選択されたproxyサーバ計算機65～67内のproxyプロセスに中継する。

【0256】本発明を非常に有効に適用できる例として、前述したインターネット上のWWWシステムがある。図9は、WWWシステム上でのredirector計算機の処理手順を示すフローチャートである。

【0257】WWWシステムではネットワーク上に分散したファイルオブジェクト名称はUniform Resource Locator(URL)と呼ばれる形式で表現され特定される。

【0258】たとえば、redirector計算機73が、クライアント計算機24からの読出要求を受信したとすると、redirector計算機73は要求に含まれるURLを取出す(S31)。取出されたURLに基づいてハッシュ関数による計算を行なう。ここで、ハッシュ関数にはさまざまなものが考えられるが、実施の形態4ではURL形式で表現されたファイルオブジェクト名称に含まれる文字の文字コードをすべて加算し、proxyサーバ計算機65から67の台数で割った剰余を使用した(S33)。

【0259】このハッシュ関数によって算出された値によって、読出要求を中継するproxyサーバ計算機65～67の番号を決定する(S34)。そして、選択されたproxyサーバ計算機65～67に読出要求が中継される(S35)。

【0260】同一のファイルオブジェクトの読出要求は、すべて同一のproxyサーバ計算機65～67で中継されることが、キャッシュの有効利用の観点からは望ましい。なぜなら、同一のファイルオブジェクトを異なるproxyサーバ計算機で重複して保持すると、キャッシュ容量が減少し、キャッシュファイルの有効利用が図れなくなるからである。そのためには、ハッシュ関数がすべてのredirector計算機72～77内のredirectorプロセスで同一のものである必要がある。

【0261】上述したハッシュ関数は、各proxyサーバ計算機65～67が均等に選択されるため、それぞれのproxyサーバ計算機65～67に対する負荷は均等になるが、適切にハッシュ関数を選ぶことにより、各proxyサーバ計算機65～67の能力に応じて意図的に均等でなく負荷を分散することも容易に実現できる。すなわち、処理能力の高いproxyサーバ計算機に対しては、選択される可能性を高くし、処理能力の低\*

[http://proxyserver:10080/\\_](http://proxyserver:10080/_)—<http://www.xxx.co.jp/test/index.html>

【0265】数23の記述のうち、最初のhttpは使用するプロトコル名を示し、proxyserverはネットワーク上のproxyサーバ計算機のアドレスを示すものである。また、10080はTCP/IP接続のポート番号と呼ばれ、proxyサーバ計算機上のどのプロセスに対し読出要求を送るのかを示す。—は、これがDeleGateの中継記法であることを示す。残りの<http://www.xxx.co.jp/test/index.html>が、本来の読出要求のURLである。

【0266】DeleGateが上記読出要求を受信すると、proxyサーバ計算機proxyserverのポート番号10080で示されるproxyプロセスに対し、URLで示されるファイルオブジェクトの読出要求を中継する。すなわち、読出要求の中で、中継するproxyサーバ計算機を指定することが可能である。

【0267】フィルタ機能は、外部プログラムを利用してDeleGate機能を拡張するための仕組みである。DeleGateはクライアント計算機もしくは他のproxyサーバ計算機からの読出要求を受信すると、受信した読出要求を一旦外部プログラムに引渡し、この外部プログラムによって処理された読出要求を再び受取って中継等の処理を行なうことが可能である。

[http://proxyserver:10080/\\_](http://proxyserver:10080/_)—<http://www.xxx.co.jp/test/index.html>

\*いproxyサーバ計算機に対しては、選択される可能性を低くすることによって各proxyサーバ計算機の処理速度を均等にするものである。

【0262】このようにして算出された番号により示されるproxyサーバ計算機が、選択されたproxyサーバ計算機となり、redirectorプロセスは選択されたproxyサーバ計算機内のproxyプロセスにクライアント計算機24からの読出要求を中継する。proxyサーバ計算機65～67内のproxyプロセスは、redirector計算機73内のredirectorプロセスにより中継されたクライアント計算機24からの読出要求を広域ネットワーク61を介してサーバ計算機11に中継し、転送されたファイルオブジェクトをキャッシュする。

【0263】redirectorプロセスは、広く一般に利用されているDeleGateを利用して実施することが可能である。実施の形態4のredirectorプロセスは、DeleGateの中継記法とフィルタ機能を利用して実現されている。以下に、中継記法の例を示す。

【0264】

【数23】

【0268】図11は、redirector計算機の内部構成を示す図である。実施の形態4において、DeleGateによって実現されるredirectorプロセス81が、たとえば、数24に示すURLを含む読出要求を受信したとする。

【0269】

【数24】

<http://www.xxx.co.jp/test/index.html>

【0270】これは、[www.xxx.co.jp](http://www.xxx.co.jp)で表わされるサーバ計算機11にこの読出要求を送信することを示している。

【0271】しかし、実施の形態4ではDeleGateのフィルタ機能により、外部プログラムとして実現されたproxy切替制御部82においてURLをもとにハッシュ関数の計算を行ない(図9のS32～S33)、その演算結果によってproxyサーバ計算機proxyserver1が選択されたとすると、上記読出要求内のURLを数25のように書替える(図9のS34)。

【0272】

【数25】

【0273】外部プログラムであるproxy切換制御部82によって書替えられた読出要求を受取ったredirectorプロセス81は、proxyserver1で表わされるproxyサーバ計算機にwww. xxx. co. jpで表わされるサーバ計算機11に対する読出要求を中継する(図9のS35)。

【0274】したがって、ハッシュ関数で計算された結果に基づいてURL中のproxyサーバ計算機のアドレスをproxyserver1に変えることにより、元の読出要求のURLに基づいて中継するproxyサーバ計算機を選択することができる。実際に、2台のproxyサーバ計算機を用意し、ハッシュ関数はURLに含まれるファイルオブジェクト名称の文字の文字コードをすべて加算し(図8のS32)、proxyサーバ計算機の台数2で割った剰余を使用した(図8のS33)。すなわち、ハッシュ関数の計算結果が0であった場合proxyserver0により中継し、1であった場合はproxyserver1で中継を行なうものとする。

【0275】図10は、図2に示す同一ページのファイルオブジェクトが、2台のproxyサーバ計算機に分散されるようすを示したものである。図10の上段は、本来の読出要求のURL、ファイルオブジェクト名称の文字コードの総計(check sum)、文字コードの総計をproxyサーバ計算機の台数である2で割った余りおよび選択されたproxyサーバ計算機のアドレスを示している。図10の下段は、proxy切換制御部82で書替られたURLを示している。この例では、ハッシュ関数の結果が0の方が多くなっており、proxyserver0への中継が多くなってしまいうが、これはサンプル数が少ないことによるものである。実際の運用では膨大なアクセスが、ハッシュ関数による分散が確率的に均等になるという性質のもとに、ほぼ2分の1ずつ均等に分散されることが確認されている。

【0276】以上説明したように、複数あるproxyサーバ計算機が別々な経路で広域ネットワークに接続されている場合、クライアント計算機からのアクセス要求を複数あるproxyサーバ計算機に分散することは、複数ある経路に分散することでもある。したがって経路の有効利用が可能となり、スループットが向上する。proxyサーバ計算機の処理能力には限界があるが、クライアント計算機からのアクセス要求が複数のproxyサーバ計算機に分散されることにより、proxyサーバ計算機1台で処理されるアクセス要求数が減少し、スループットが向上する。

【0277】また、同一ページ内に複数のファイルオブジェクトが含まれている場合、それらのファイルオブジェクトに対するアクセス要求も複数のproxyサーバ計算機によって分散して処理されるので、単一ページのアクセス速度も向上する。

【0278】また、どのクライアント計算機からの中継要求であっても、同一URLに対するハッシュ関数が同じであるので、常に同じproxyサーバ計算機が選択されることになり、同一URLのアクセスが2回以上あればキャッシュによるアクセスの高速化が図れる。さらに、ハッシュ関数が一意にきまるものであれば任意の関数が選択できるので、proxyサーバ計算機の数もまた任意の台数が使用できる。

【0279】実施の形態4において、ハッシュ関数内で使われているURLに含まれる文字の文字コードをすべて加算した値(check sum)は、平均して数千程度になることが確かめられている。たとえば、シャープ奈良工場における実施においては、1996年3月13日に実測された30481件のファイルオブジェクトに対するアクセス要求のURLに含まれる文字や文字コードの合計の平均は4438(小数点以下四捨五入)であった。

【0280】前述したとおり、実施の形態4ではこのアクセス要求のURLに含まれる文字や文字コードの合計をproxyサーバ計算機の台数で割った剰余をハッシュ関数として使用し、proxyサーバ計算機の使用している。したがって、図10の例ではproxyサーバ計算機は2台であるが、この台数が数千程度であっても、ハッシュ関数によるproxyサーバ計算機の使用はほぼ均等に行なわれると考えられる。

【0281】また、同一のハッシュ関数であれば、redirector計算機の数も任意である。したがって、本方式ではスケーラビリティも非常に優れている。

【0282】また、本方式ではキャッシュ付のproxyサーバ計算機は既存のものをそのまま使用することが可能である。したがって、図24の従来の方式に比べ導入が容易で経済的である。また、クライアント計算機に関しても、既存のクライアント計算機を使用して、redirector計算機を既存のproxyサーバ計算機に見立てて設定を行なうだけでよく、特殊なクライアント計算機を用意する必要はない。

【0283】[実施の形態5] 実施の形態5は、実施の形態4において独立した計算機上で実行していたredirectorプロセスを、クライアント計算機24上で実行する。図12は、実施の形態5における装置の内部構成を示す図であり、クライアント計算機24内のクライアントプロセス84(図8のクライアント計算機24と同じ動作をするプロセス)からの読出要求は、ローカルネットワーク上には送出されず、クライアント計算機24内部のredirectorプロセス85に直接送信される。この送信は、たとえばUNIXのメッセージ通信によって行なわれる。

【0284】redirectorプロセス85がアクセス要求を受信した後の処理は、上述したredirector計算機72~77内で動作するredirec

torプロセス81と同様であり、ローカルネットワークを介する他のクライアント計算機からのアクセス要求を受信して処理することも可能である。したがって、ここでの詳細な説明は繰返さない。また、このredirectorプロセス85の機能をクライアントプロセス84のプログラムに直接組込むことも可能である。

【0285】図8に示す実施の形態4のクライアント計算機24とredirector計算機72～77との通信が、実施の形態5では同じクライアント計算機24内部で処理されるため、ローカルネットワーク上での通信量が削減できる。したがって、独立したredirector計算機を用意する必要がなく、クライアント計算機24を利用してredirector計算機の機能を実現できるためコストの削減が可能となる。

【0286】〔実施の形態6〕実施の形態6では、実施の形態4において独立した計算機上で実行していたredirectorプロセスを、proxyサーバ計算機上で実行する。図13は、実施の形態6における装置の内部構成を示す図であり、たとえば、クライアント計算機24bから読出要求があった場合に、その読出要求はproxyサーバ計算機99内のredirectorプロセス91で受信される。その後、redirectorプロセス91は、図9を用いて説明したようにアクセス要求を中継するproxyサーバ計算機を選択し、その選択されたproxyサーバ計算機が98であれば、proxyサーバ計算機98のproxyプロセス86に対してアクセス要求を中継する(92)。また、受信したredirectorプロセスの動作しているproxyサーバ計算機と選択されたproxyサーバ計算機がともに99である場合、読出要求の中継はネットワーク上に送出されずproxyサーバ計算機99内部で直接行なわれる(93)。そして、proxyプロセス(86または89)が広域ネットワーク61を介してサーバ計算機11に対してアクセス要求を送出する(62または63)。

【0287】同様に、クライアント計算機24aから読出要求があった場合に、その読出要求は、proxyサーバ計算機98上のredirectorプロセス96で受信される。その後、redirectorプロセスは、proxyサーバ計算機を選択を行ない、その選択されたproxyサーバ計算機が99であれば、proxyサーバ計算機99のproxyプロセス89に対してアクセス要求を中継する(95)。

【0288】また、受信したredirectorプロセスの動作しているproxyサーバ計算機と選択されたproxyサーバ計算機がともに98である場合、読出要求の中継はネットワーク上に送出されずproxyサーバ計算機98内部で直接行なわれる(96)。そして、proxyプロセス(86または89)が広域ネットワーク61を介してサーバ計算機11に対してアクセ

ス要求を送出する(62または63)。

【0289】サーバ計算機11から取得したファイルオブジェクトをキャッシュファイル(87または90)に格納する処理は、上述した処理と同様であるので詳細な説明は繰返さない。

【0290】また、このredirectorプロセスの機能を、proxyサーバ計算機上でproxyサーバとしての機能を実現しているプログラムに直接組込むことも可能である。

【0291】したがって、redirector計算機とproxyサーバ計算機の通信の一部がproxyサーバ計算機内部で処理されるため、ローカルネットワーク上での通信量が削減できる。また、独立したredirector計算機を用意する必要がなく、proxyサーバ計算機を利用してredirector計算機の機能を実現できるためコストを削減することが可能になる。

【0292】〔実施の形態7〕実施の形態4のredirector計算機72～77はアクセス要求の中継のみを行っていた。しかし、図23で説明したようなツリー状の構成が有効であるような大規模なローカルネットワークでは、1次キャッシュproxyサーバ計算機503～508の持つキャッシュ機能をredirector計算機に持たせることによって、ネットワーク上の通信量およびproxyサーバ計算機の負荷を軽減することが可能となる。図14は、そのシステム構成を示す図である。redirectorサーバ計算機40～45は、図11に示すredirector計算機80と図16に示すproxyサーバ計算機13を組合せることにより可能である。すなわち、図16のリード要求17を図11のredirectorプロセス81が受信することによって実現可能であるので、ここでの詳細な説明は繰返さない。また、実施の形態5におけるクライアント計算機内部でキャッシュを行なうことによって、ローカルネットワーク上の通信量の削減とproxyサーバ計算機の負荷を軽減することが可能となる。

【0293】上述したように、redirector計算機においてキャッシュすることにより、ローカルネットワーク上での通信量およびproxyサーバ計算機の負荷を軽減することができる。

【0294】〔実施の形態8〕上述した実施の形態4から7は、排他的に使用されなければいけないものではない。すなわち、実施の形態8においてはredirector機能を実現するすべての計算機(proxyサーバ計算機、クライアント計算機、redirector計算機)において、redirector機能の使用するハッシュ関数が同一であり、すべてのアクセス要求がそれぞれ同一のproxyサーバ計算機等のキャッシュ機能を有する計算機で中継およびキャッシュされれば、実施の形態4～7に記載された各種の計算機は、単一の

ネットワーク内に混在して使用されても全く問題はない。したがって、それぞれの会社等の組織におけるさまざまなネットワークや計算機の運用事情に応じ、非常に柔軟なネットワーク運用を行なうことが可能となる。

【0295】ある程度以上の大規模なローカルネットワークでは、広域ネットワークとの接続が複数の経路（図8の62～64）からなる場合もある。そのような場合、proxyサーバ計算機ごとに異なる経路を使用すると、クライアント計算機からのアクセス要求を複数ある経路に分散して中継することになる。したがって、複数ある広域ネットワークとの通信経路を有効利用し、広域ネットワークとローカルネットワークとの通信のスループット向上が図れる。

【0296】実際の運用においては、すべての計算機の設定を一度に変えることは現実的ではない。移行には暫く時間がかかることが普通である。より効率のよい運用を図るためには、すべてのredirector機能を有する計算機内でハッシュ関数を一致させる必要があるため、すべてのクライアント計算機がredirector計算機を利用するようシステムを構築するか、クライアント計算機がredirector機能を持つ必要がある。しかし、そのようなシステムを構築しなくても\*

\*redirector機能を有する計算機を使用することは可能である。すなわち、本発明の各方式では、従来のproxyサーバ計算機やクライアント計算機を全く手を付けずに状態で、システムの一部にredirector機能を持たせて、システム全体はそのまま使用することも可能である。

【0297】同様に、システムの再構築の際、複数のredirector機能におけるハッシュ関数が一致しない場合もあり得る。このような場合でも、システム全体の動作自体には影響はないので、通常と同様に利用が可能である。上述したいずれの場合も利用は可能であるが、proxyサーバ計算機その他の有効利用の観点からは望ましい状況とはいえない。したがって、できるだけ早い時期にすべてのredirector機能を有する計算機での設定を統一させることが望ましい。しかし移行期に新旧の設定を混在していても不必要な混乱を起こさず、スムーズに移行することが可能である。

【0298】なお、httpプロトコルを含めた参考文献リストを以下に示す。

【0299】

【数26】

・[Ref. 1]横塚実、『インターネットとはなにか』

[http://www.rwcp.or.jp/people/yokotuka/internet\\_report.html](http://www.rwcp.or.jp/people/yokotuka/internet_report.html)

・[Ref. 2]page170, "キャッシュ利用でトラフィックを低減するプロキシサーバー"

『企業ユーザのためのインターネットハンドブック』日経コミュニケーション、

ISBN4-822-1365-X, 日経BP社

・[Ref. 3]WWWに関する参考文献

<http://www.w3.org/hypertext/WWW/Protocols/Overview.html>

・[Ref. 4]HTTPの仕様に関するインターネット上の文献

<http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html>

・[Ref. 4の日本語資料]日本語のHTTPプロトコル仕様書としては、木内貴弘

(kiuchi@epistat.m.u-tokyo.ac.jp)氏翻訳の

[http://www.epistat.m.u-tokyo.ac.jp/internet/draft-fielding-http-spec-01-](http://www.epistat.m.u-tokyo.ac.jp/internet/draft-fielding-http-spec-01-jp.txt)

[jp.txt](http://www.epistat.m.u-tokyo.ac.jp/internet/draft-fielding-http-spec-01-jp.txt)、オリジナルファイル名:著者:T.Berners-Lee、R.T.Fielding、H.Frystyk

Nielsenがある。

・[Ref. 5]電子技術総合研究所の佐藤氏の作成されたdelegateについては

<http://www.etl.go.jp:8080/etl/People/ysato@etl.go.jp/DeleGate/> から情報が入手できる。

・[Ref. 6]松下電器産業株式会社、特開平6-290090、"遠隔ファイルアクセス

システム

・[Ref. 7]WWWシステム全般の情報は、World Wide Webコンソーシアム

<http://www.w3.org/hypertext/WWW/TheProject/>

- ・[Ref. 8]CERN httpdに関しては  
<http://www.w3.org/hypertext/WWW/Daemon/>
- ・[Ref. 9]大家隆弘(徳島大学), WWW 多段proxy計画(案)  
<http://www.orions.ad.jp/project/multiproxy/multiproxy-jp.html>
- ・[Ref. 10]CERNhttpdでのキャッシュ制御解説  
<http://info.cern.ch/hypertext/WWW/Daemon/User/Config/Caching.html>
- ・[Ref. 11]東田学, 対外接続のトラフィックを軽減するためのキャッシュ・サーバ導入  
について <http://www.osaka-u.ac.jp/odins/document/proxy-jp.html>
- ・[Ref. 12]佐藤豊, "多目的プロキシ・サーバ" eleGateの機能詳細" インターフェイス  
1995年9月p130-p146
- ・[Ref. 13]インターナショナル・ビジネス・マシーンス・コーポレーション、特開平6-309264、  
"キャッシュ・サーバ・ノードを有するコンピュータ・ネットワークにおける資源を探索する方法  
及び装置"
- ・[Ref. 14]"Siva——A hierarchical caching HTTP proxy"  
<http://www.cs.cornell.edu/Info/People/mdw/siva/siva.html>
- ・[Ref. 15]"A Central Caching Proxy Server for WWW users at  
the University of Melbourne"  
<http://www.scu.edu.au/ausweb95/papers/indexing/ocallaghan/>
- ・[Ref. 16]Lynx ブラウザについて  
[http://kufacts.cc.ukans.edu/about\\_lynx/about\\_lynx.html](http://kufacts.cc.ukans.edu/about_lynx/about_lynx.html)

#### 【図面の簡単な説明】

【図1】本発明におけるゲートウェイ装置の一例である proxyサーバ計算機の動作概念図である。

【図2】WWWサーバ内の情報ページをクライアント計算機に読出したときの表示画面を示す図である。

【図3】実施の形態1のゲートウェイ装置である proxyサーバ計算機で行なわれるキャッシュ更新プロセスの処理手順を示すフローチャートである。

【図4】proxyプロセスにおけるキャッシュファイル管理の処理手順を示すフローチャートである。

【図5】実施の形態2のゲートウェイ装置である proxyサーバ装置で行なわれるキャッシュ更新プロセスの処理手順を示すフローチャートである。

【図6】実施の形態3のゲートウェイ装置である proxyサーバ装置で行なわれるキャッシュ更新プロセスの処理手順を示すフローチャートである。

【図7】本発明を実施した場合のサーバ計算機とクライアント計算機の間で転送されるデータのタイムチャートである。

【図8】実施の形態4における分散ファイルシステムの全体構成を示す図である。

【図9】実施の形態4における redirector 計算機の処理手順を示すフローチャートである。

【図10】図2に示す同一ページのファイルオブジェクトに対するアクセス要求が2台のサーバ計算機に分散されるようすを示す図である。

【図11】実施の形態4における redirector 計算機の内部構成を示す図である。

【図12】実施の形態5におけるクライアント計算機の内部構成を示す図である。

【図13】実施の形態6における proxyサーバ計算機の内部構成を示す図である。

【図14】実施の形態7における分散ファイルシステムの全体構成を示す図である。

【図15】従来のゲートウェイ計算機の構成を示す図である。

【図16】proxyサーバ装置の概念図である。

【図17】proxyサーバ装置の回路構成の第1の例を示す図である。

【図18】proxyサーバ装置の回路構成の第2の例を示す図である。

【図19】proxyサーバ装置の回路構成の第3の例であって、本願発明の実施の形態で proxyサーバ装置としても使用される proxyサーバ装置の回路構成を示す図である。

【図20】キャッシュヒット率と平均アクセス速度との関係を示すグラフである。

【図21】キャッシュファイルサイズとキャッシュヒット率との関係を示すグラフである。

【図22】キャッシュ有効期限とキャッシュファイルの大きさとの関係を示すグラフである。

【図23】従来の分散ファイルシステムの全体構成の第



57

1の例を示す図である。

【図24】従来の分散ファイルシステムの全体構成の第2の例を示す図である。

【図25】従来の分散ファイルシステムの全体構成の第3の例を示す図である。

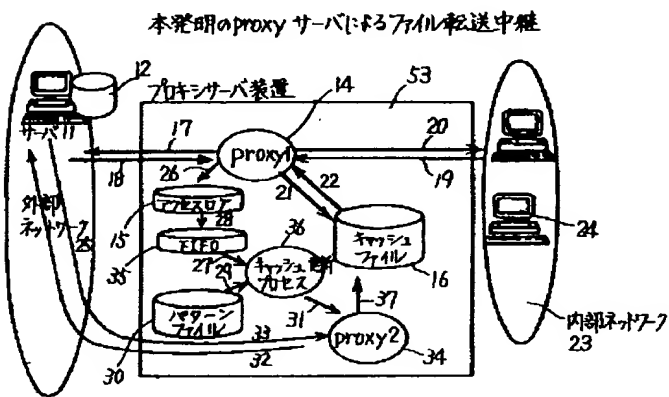
【図26】従来の分散ファイルシステムにおけるサーバ計算機内のデータの更新とクライアント計算機が読出したデータとの関係を示すタイムチャートである。

【符号の説明】

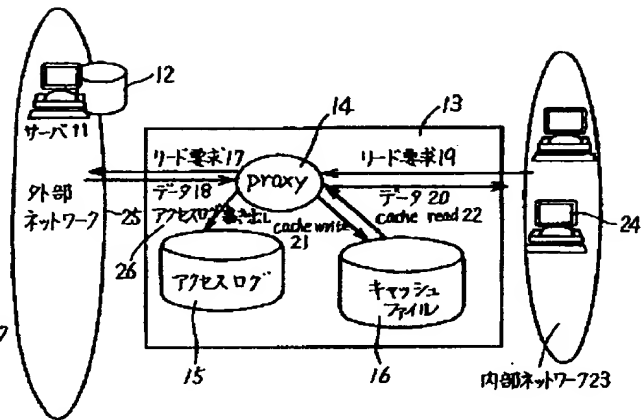
58

- 11 サーバ計算機
- 14 第1のproxyプロセス
- 15 アクセスログ
- 16 キャッシュファイル
- 30 パターンファイル
- 34 第2のproxyプロセス
- 35 FIFOバッファ
- 36 キャッシュ更新プロセス
- 72~77 redirector計算機

【図1】



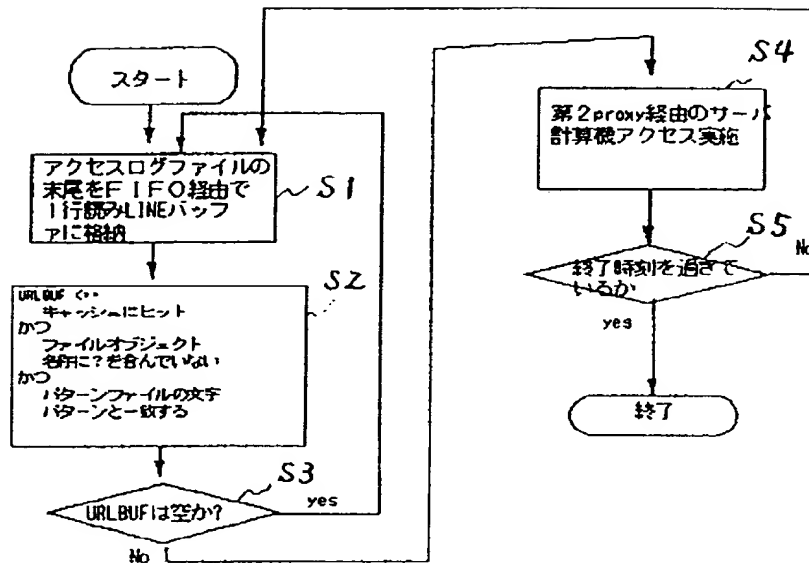
【図16】



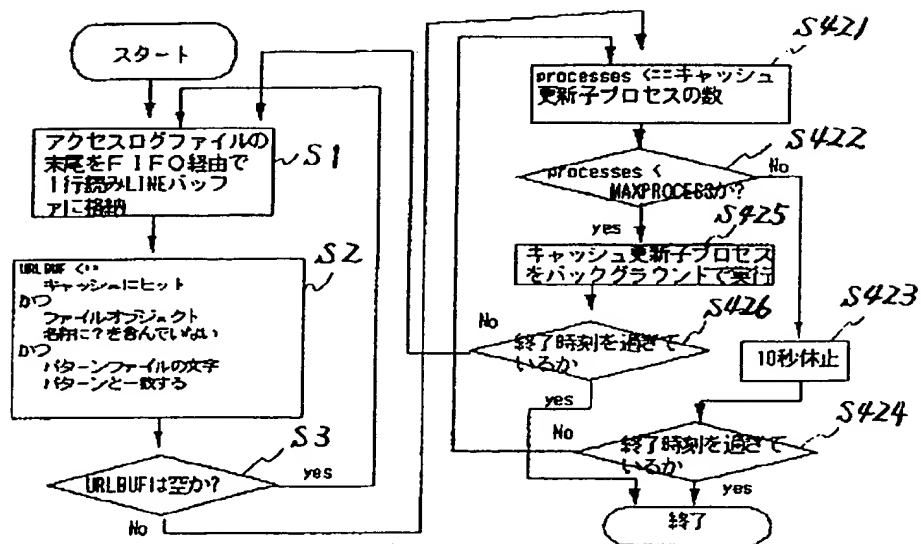
【図2】



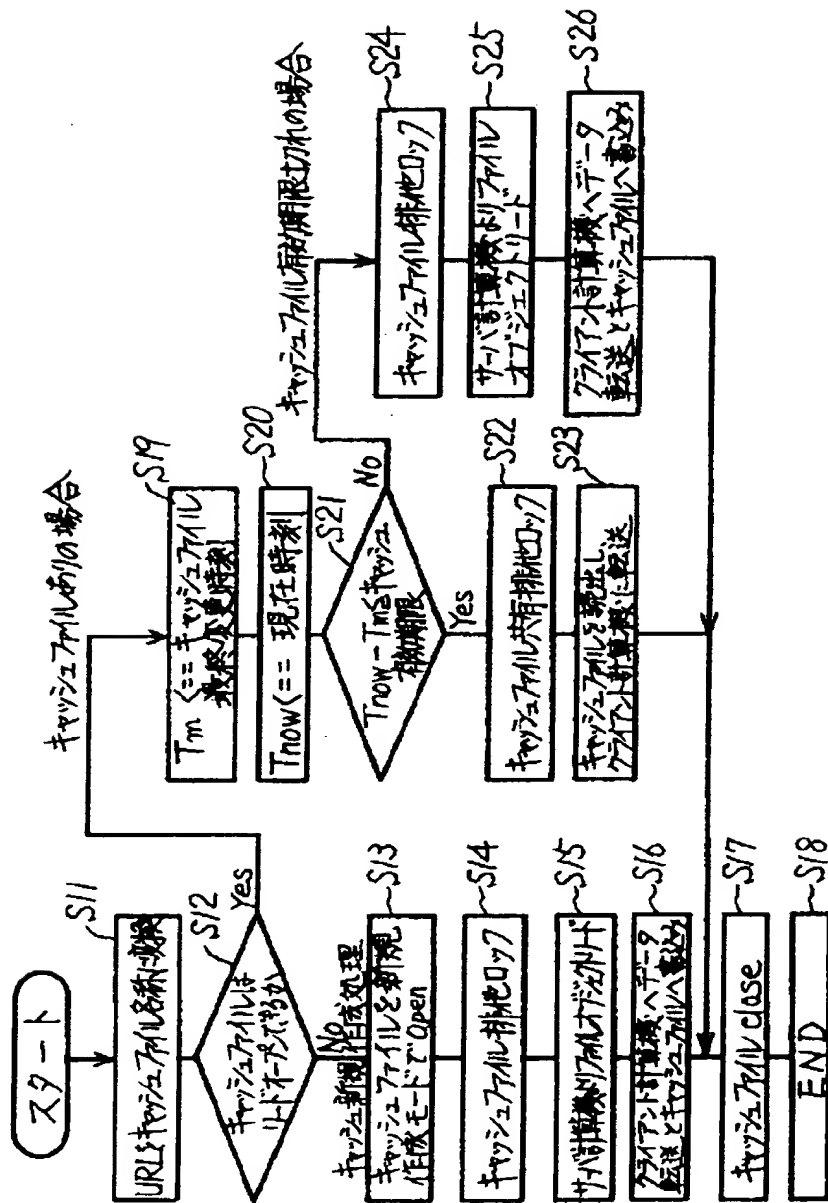
【図3】



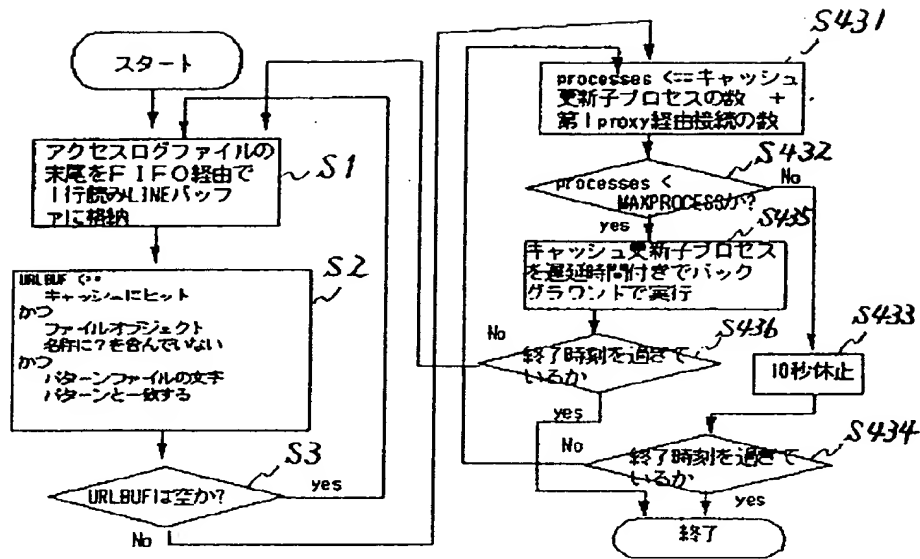
【図5】



【図4】

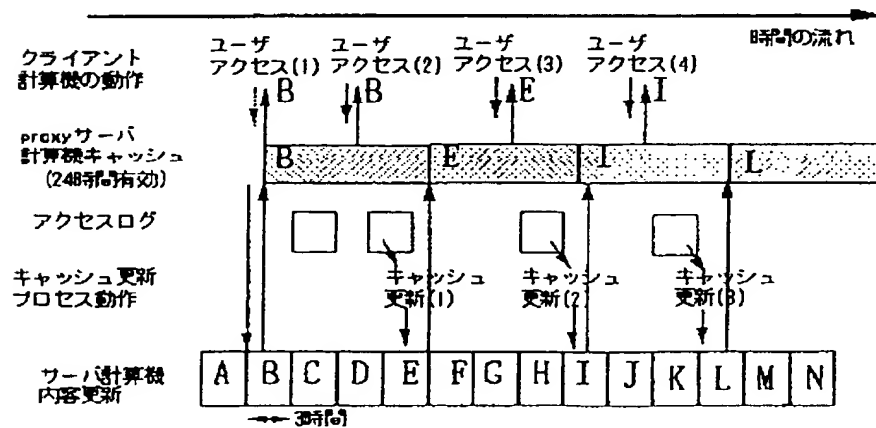


【図6】

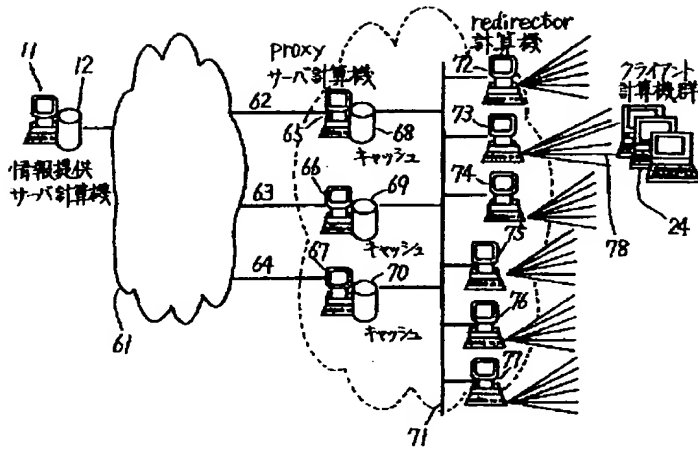


【図7】

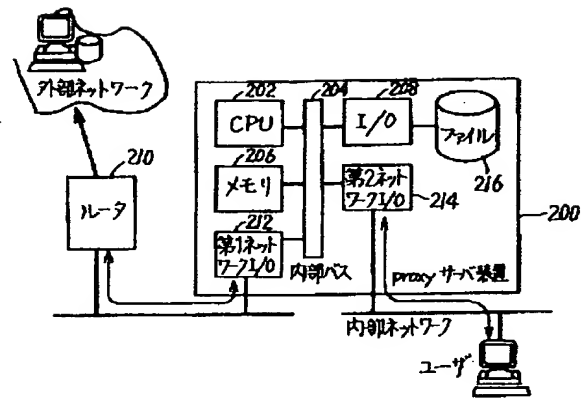
### 変更間隔3時間のサーバデータアクセスのタイムチャート (キャッシュ更新プロセスの動作時)



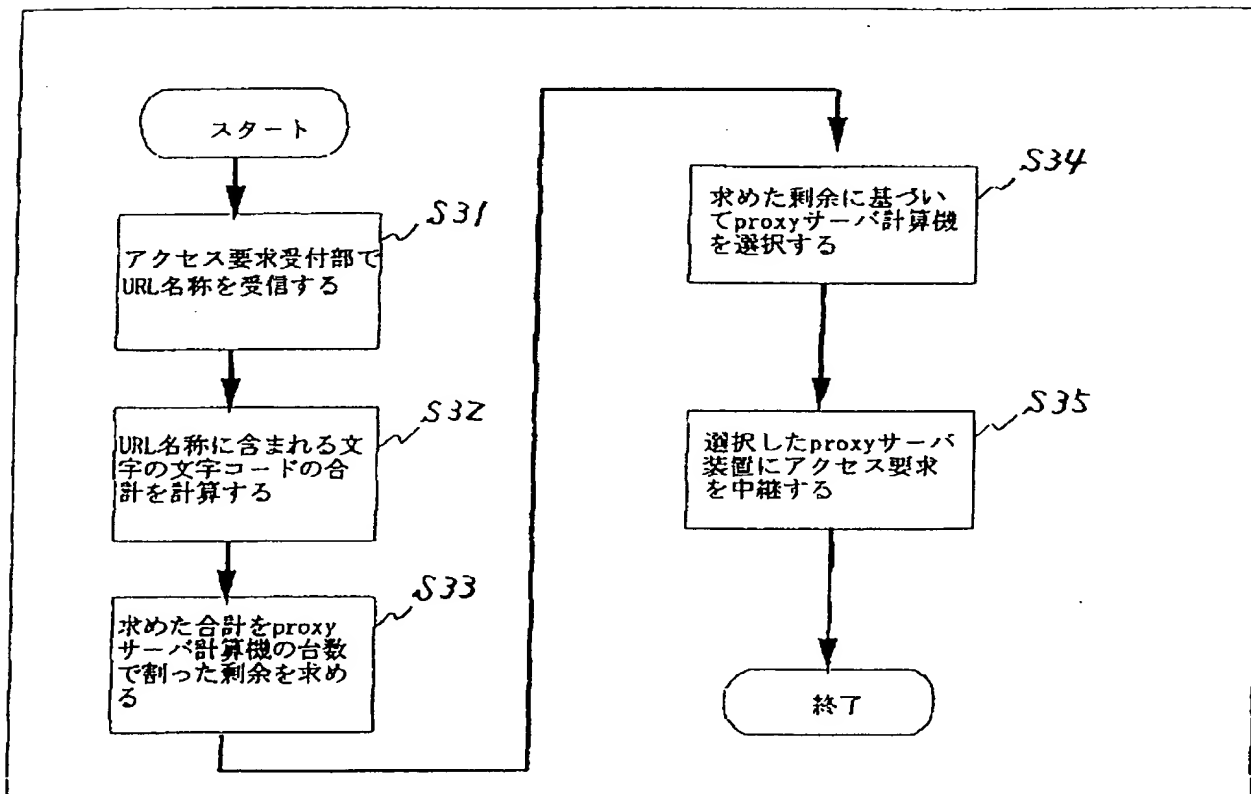
【図8】



【図17】



【図9】



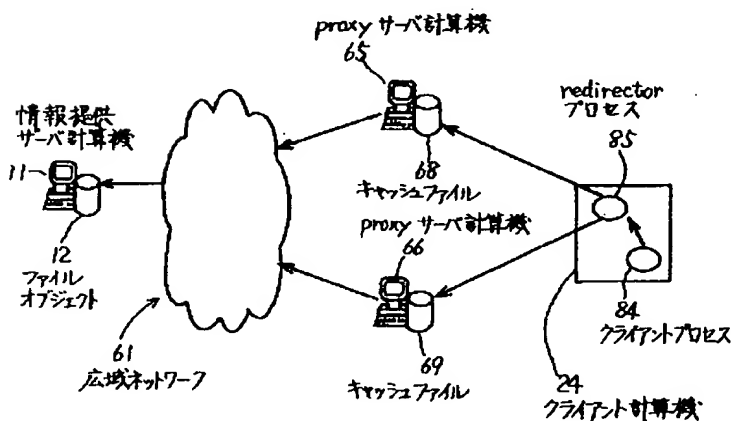
【図10】

URL	check sum	checksumの 2の剰余	選択された proxyserver
http://naragw.sharp.co.jp/	2395	1	proxyserver1
http://naragw.sharp.co.jp/sharpcolor.gif	3836	0	proxyserver0
http://naragw.sharp.co.jp/isg.gif	3974	0	proxyserver0
http://naragw.sharp.co.jp/zaurus.gif	3401	1	proxyserver1
http://naragw.sharp.co.jp/Shoin.gif	3264	0	proxyserver0
http://naragw.sharp.co.jp/Prostation.gif	3826	0	proxyserver0
http://naragw.sharp.co.jp/S2.gif	2884	0	proxyserver0

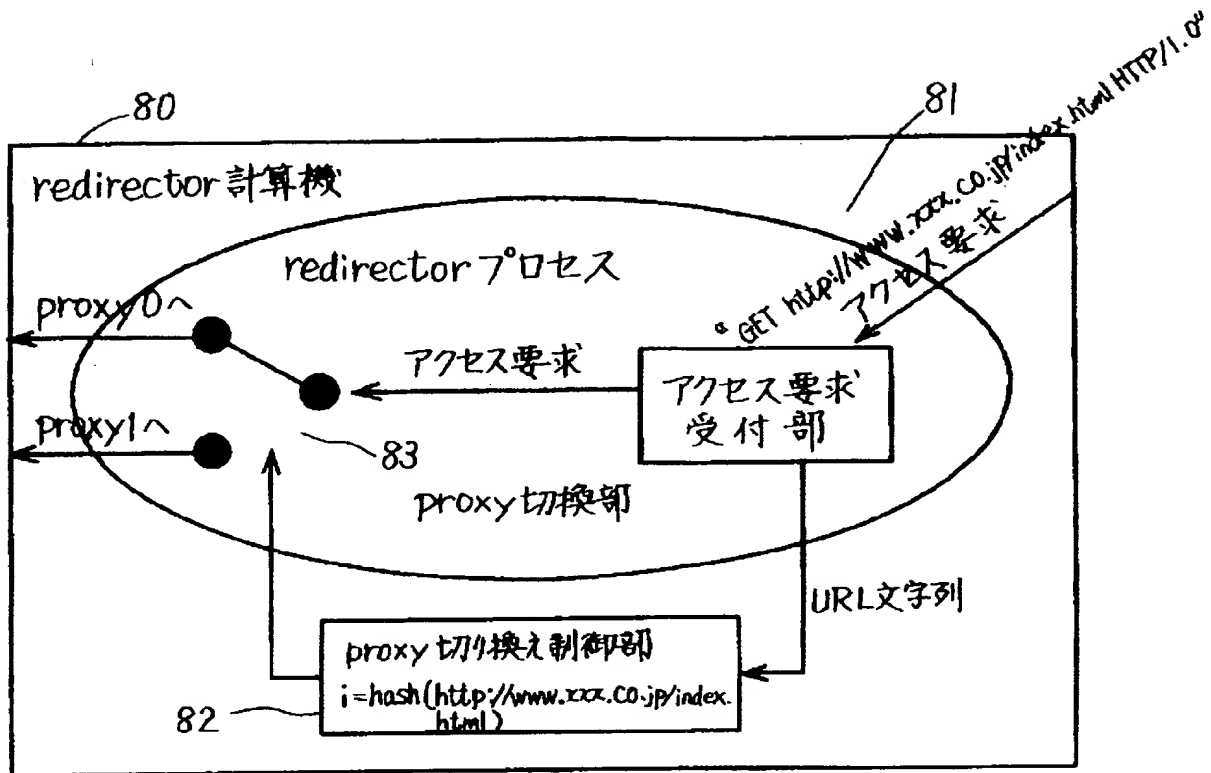
  

変換後のURL
http://proxyserver1:10080/_-http://naragw.sharp.co.jp/
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/sharpcolor.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/isg.gif
http://proxyserver1:10080/_-http://naragw.sharp.co.jp/zaurus.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/Shoin.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/Prostation.gif
http://proxyserver0:10080/_-http://naragw.sharp.co.jp/S2.gif

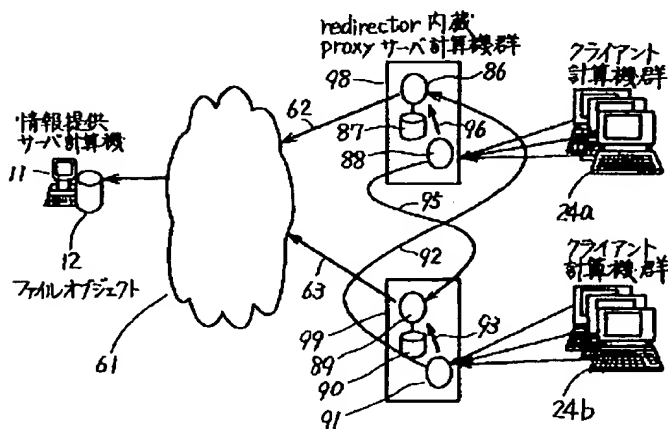
【図12】



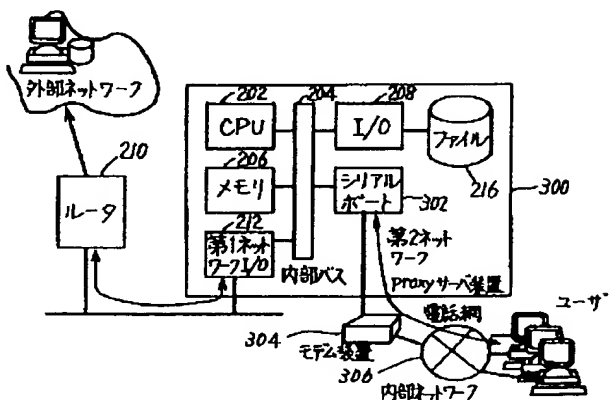
【図11】



【図13】

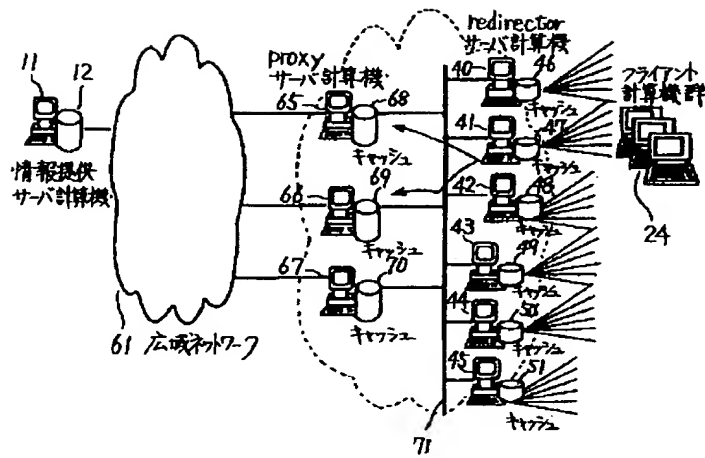


【図18】

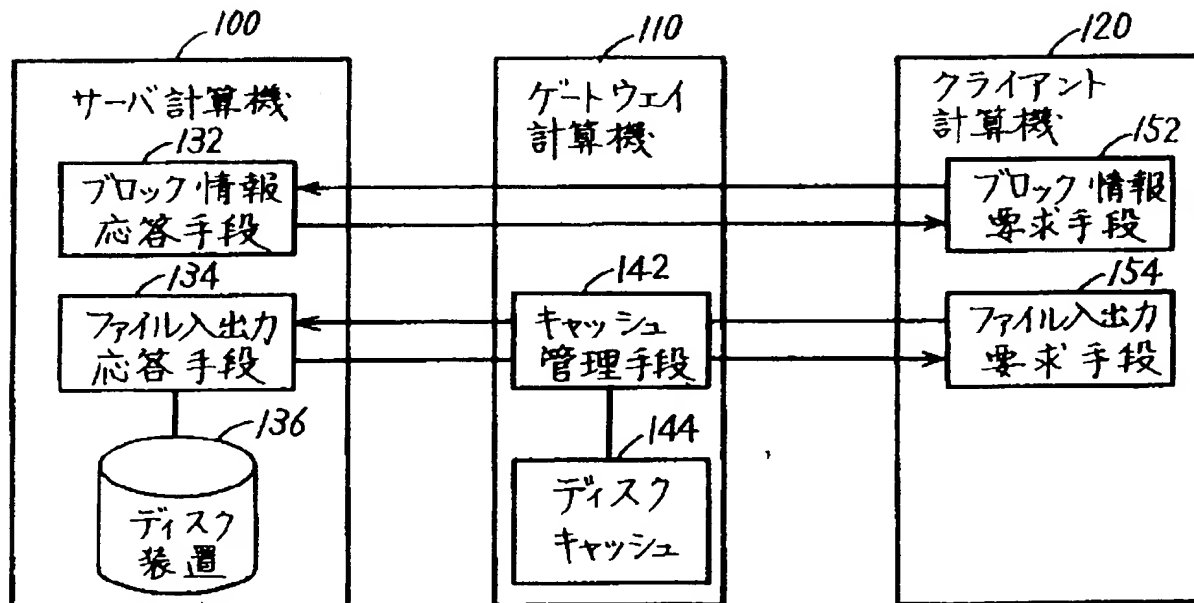




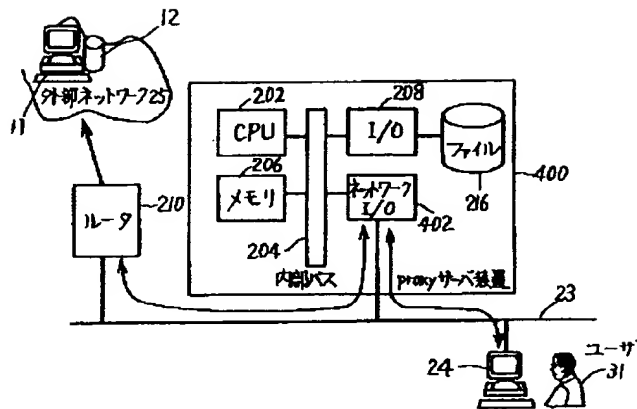
【図14】



【図15】



【図19】

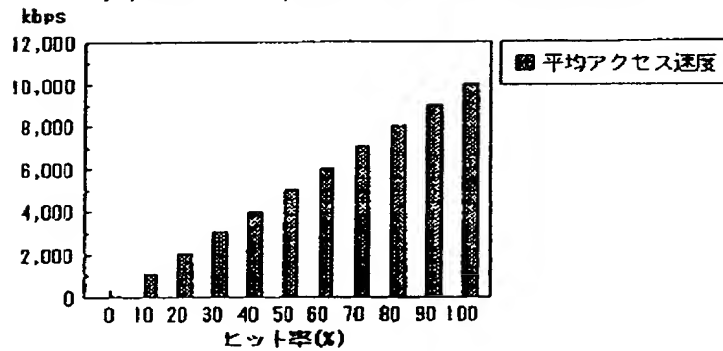


【図20】

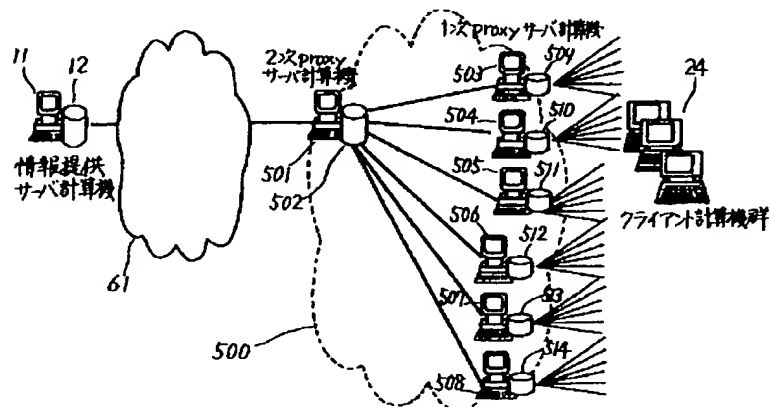
## キャッシュヒット率向上による 平均アクセス速度向上

$$\text{平均アクセス速度} = 64\text{kbps} \times \text{ミスヒット率} + 10\text{Mbps} \times \text{ヒット率}$$

キャッシュヒット率と平均アクセス速度の関係

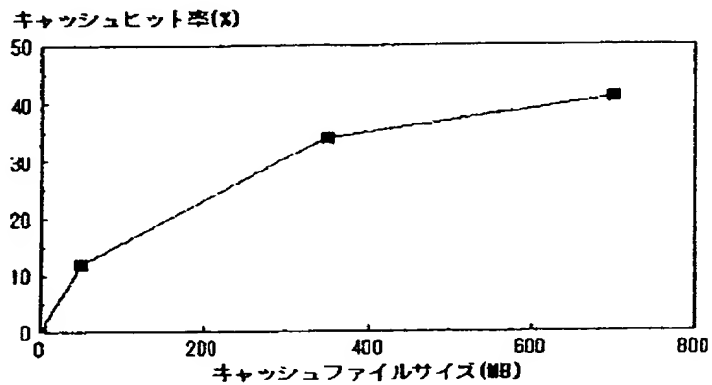


【図23】



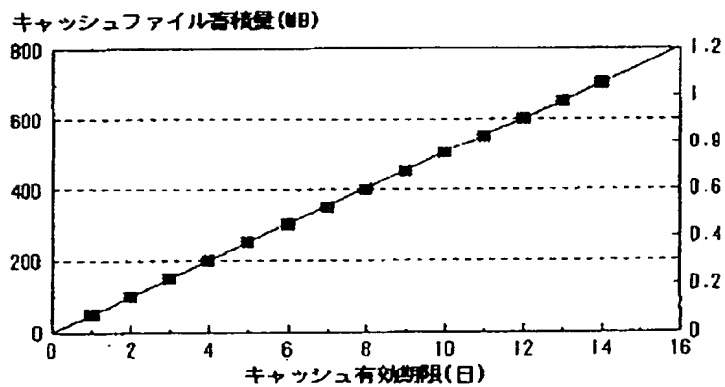
【図21】

## キャッシュファイルサイズと キャッシュヒット率の関係

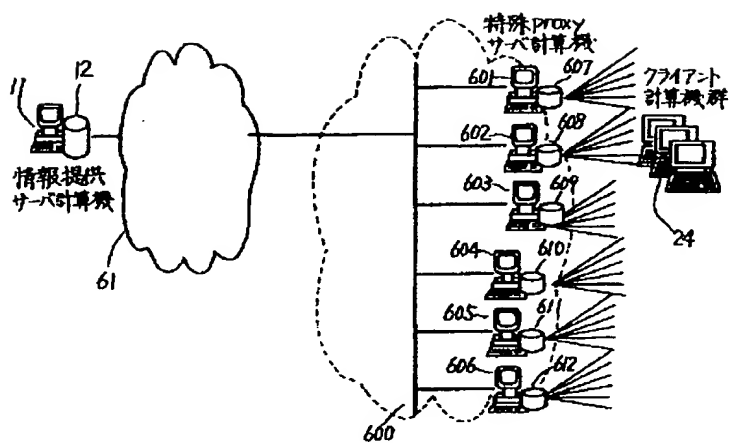


【図22】

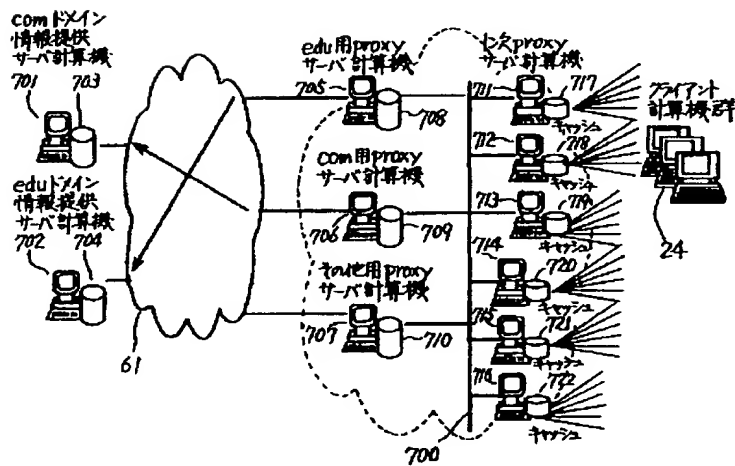
## キャッシュ有効期限とキャッシュ ファイルの大きさの関係



【図24】



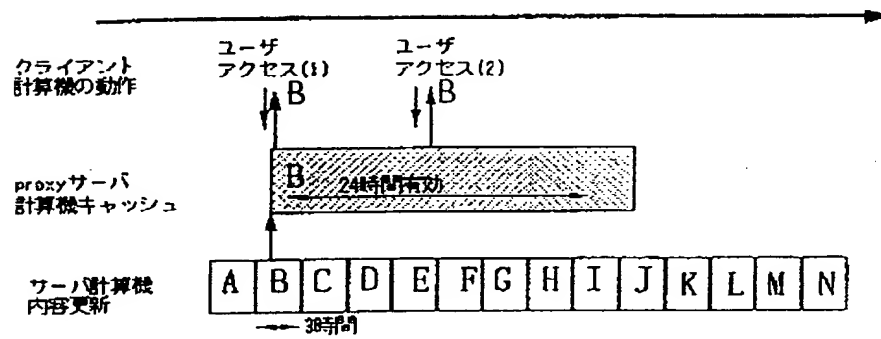
【図25】



【図26】

## 変更間隔3時間のサーバデータアクセスのタイムチャート

- proxyサーバ計算機でのキャッシュは有効期限24時間
- サーバ計算機のファイルオブジェクトは3時間間隔で更新



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☒ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**